UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

Diseño e implementación de un vehículo a escala controlado remotamente

---

Trabajo fin de grado

Madrid, septiembre de 2016

Autor: Víctor Torrico López

Tutor: Javier Fernández Muñoz

Resumen

Este proyecto describe el diseño y la implementación de un vehículo controlado de manera remota a través de un dispositivo móvil. Estará formado por dos grandes bloques: hardware y software. El prototipo de vehículo, construido a escala mediante una estructura que contiene un microcontrolador, podrá desplazarse con las órdenes que el usuario envíe o de manera automática, gracias a la creación de un algoritmo inteligente que evita las colisiones con objetos frontales. En el presente documento se detalla todo el proceso llevado a cabo en la realización del proyecto.

Abstract

This project describes the design and the implementation about a remote control vehicle which is moved through a mobile phone. It consists of two blocks: hardware and software. The prototype vehicle, which is built to scale through a structure that contains a microcontroller, could move with the orders from the user or automatically thanks to the creation of an intelligent algorithm which does not allow collisions to front objects. The current document will be detail all the process conducted in the realization of the project.

Índice de contenidos

Índice de ilustraciones

Índice de tablas

Capítulo 1: Introducción

En este capítulo se ofrecerá una visión general de la tecnología que se utilizará en el proyecto, así como la motivación principal y los objetivos que se deberán cumplir. Además, se indicarán las fases que tendrá el proyecto y, para facilitar la lectura del documento, se presentará la estructura que contendrá.

1.1. Visión general y motivación

En la actualidad la tecnología móvil se encuentra en constante evolución; cada vez son mayores las utilidades que puede ofrecer un dispositivo móvil, desde utilidades básicas como son realizar llamadas, a utilidades más complejas.

El impacto que genera en la sociedad el auge de la tecnología móvil, puede reflejarse en el número de terminales vendidos por año entre personas de todas las edades; esto despierta un interés tanto en los usuarios como en los desarrolladores de aplicaciones, ya que cada vez existe un mayor número de aplicaciones destinadas al ocio o para facilitar ciertas tareas al usuario.

De aquí surge la motivación fundamental por la cual se va a realizar este proyecto, que consiste en aplicar la tecnología del radio control a la tecnología móvil. Los coches teledirigidos fueron una gran revolución tecnológica; el poder manejar un coche con un mando de forma inalámbrica despertó gran interés en el mundo del ocio. En la actualidad existen estructuras teledirigidas más complejas, como barcos, aviones, helicópteros o los innovadores “drones”.

El poder unir el concepto de radio control con la tecnología móvil, es el principal objetivo de este proyecto, que consistirá en un prototipo de coche que podrá ser dirigido a través de un teléfono inteligente o “smartphone”.

1.2. Objetivos

El principal objetivo del proyecto es controlar un vehículo a escala de manera remota mediante un teléfono móvil, y que además pueda desplazarse de manera autónoma mediante un simple algoritmo.

Para poder llevar a cabo el objetivo principal, será necesario cumplir una serie de sub-objetivos que se muestran a continuación:

- Diseño y construcción mecánica de una estructura que simule un vehículo.

La estructura deberá contener lo básico para realizar las funcionalidades de un vehículo, dado que en este proyecto se pretende el diseño e implementación de un prototipo, que posteriormente podrá adquirir la forma de un vehículo real con más componentes.

- Implementación de un software para un controlador que controle los posibles movimientos del vehículo.

Para que el vehículo se pueda desplazar, son necesarias dos partes fundamentales: un controlador y un software que maneje el vehículo.

- Desarrollo de una aplicación para teléfono móvil que permita al usuario el manejo libre del vehículo.

Puesto que la idea fundamental es realizar un control remoto distinto al habitual, se implementará mediante una aplicación móvil, que servirá de mando e incluirá todos los controles para manejar el vehículo.

- Creación de un modo manual y automático.

El modo manual será el modo en el que el usuario controle el vehículo, y el modo automático en el que se realice el movimiento autónomo. Este último además incluirá un sistema anti-colisiones frontales.

Para cumplir estos sub-objetivos, se deben tener en cuenta además una serie de consideraciones:

- La aplicación de control remoto debe ser sencilla e intuitiva, de modo que pueda ser utilizada por cualquier usuario sin ningún problema ni dificultad.

- Se debe crear una conexión inalámbrica entre el vehículo y el dispositivo móvil con el que se maneje.

- El hardware del vehículo deberá contar con una fuente de alimentación externa que permita la máxima movilidad posible.

- Tanto el software del controlador como el de la aplicación móvil deben coordinarse correctamente para que el vehículo procese las órdenes sin errores.

Por lo tanto, este proyecto consta de dos grandes bloques, cuya implementación y diseño podrán realizarse por separado, pero finalmente deben funcionar en conjunto:

- Hardware del sistema: incluye todas las componentes que forman la estructura del vehículo y todos los controladores y chips que pueda contener.

- Software del sistema: estará formado por la aplicación de manejo para el dispositivo móvil y por el programa cargado en el controlador.

1.3. Fases de desarrollo

La primera fase consiste en determinar un plan de propuesta de desarrollo y los objetivos que se deberán cumplir.

Una vez definidos los objetivos del proyecto, se establece una planificación estimada de las tareas y un presupuesto inicial; durante la realización del proyecto, tanto la planificación como el presupuesto fijados se podrán ver alterados debido a un imprevisto o a las modificaciones que puedan surgir.

La tercera fase consiste en una investigación y un estudio de la situación de la tecnología actual y de los productos existentes, para así poder decidir qué utilizar en el desarrollo del proyecto. Esta tarea se denomina estudio de viabilidad del sistema. Una vez determinado todo lo que se utilizará en el proyecto, se puede pasar a la siguiente fase.

La cuarta fase consta de dos partes; primero una entrevista con el cliente para conocer sus necesidades en cuanto al desarrollo del sistema, y posteriormente la elaboración de un plan de requisitos en el que tener en cuenta, tanto los requisitos impuestos por el cliente o requisitos de usuario, como los requisitos que el sistema deberá cumplir.

Una vez finalizada la fase de análisis, se procede al diseño del sistema, en el que se definirá la arquitectura tanto hardware como software de un modo esquemático, para su posterior implementación.

La implementación es la fase de programación del software a nivel de código, y de la construcción de todas las componentes para formar una estructura hardware. Esta tarea es la que conllevará un mayor número de horas.

Finalmente, una vez implementado el sistema, se realiza una última fase de pruebas, donde se tendrán en cuenta los requisitos establecidos y el resultado final de la implementación, para realizar todas las pruebas y verificar que el sistema funciona de forma correcta.

La elaboración de la documentación, tanto de esta memoria como de los anexos que pueda incluir, se realizará en una última fase.

1.4. Estructura del documento

El proyecto estará estructurado en una serie de capítulos, cuyo resumen del contenido se explica a continuación:

Capítulo 1: Introducción

En este capítulo se explicará la motivación que ha conllevado la realización de este proyecto y los objetivos que se deberán cumplir. Además, se incluirá un glosario de términos y acrónimos para facilitar a cualquier usuario inexperto en la materia la lectura y comprensión de este documento.

Capítulo 2: Estado de la cuestión

El estado de la cuestión, también denominado "estado del arte", explica la situación actual de la tecnología utilizada para el proyecto, y las alternativas que existen.

Capítulo 3: Análisis del sistema

En este capítulo se establecen los requisitos que deberá cumplir el sistema acorde a las necesidades del cliente. También se definirán los casos de uso posibles en base a los requisitos acordados con el cliente, así como la especificación técnica de dichos requisitos.

Capítulo 4: Diseño e implementación del sistema

El objetivo de este capítulo es detallar el diseño que se ha llevado a cabo para desarrollar tanto el software como el hardware del sistema, y la implementación del diseño realizado.

Capítulo 5: Pruebas del sistema

En este capítulo se realizará una planificación de las pruebas que serán llevadas a cabo para verificar si el sistema funciona o no correctamente, y detectar posibles errores en el diseño o en la implementación.

Capítulo 6: Planificación y presupuesto

En esta parte se especificará una planificación estimada de las tareas que se van a realizar en el proyecto, así como la duración posible que tendrá cada una.

También se realizará un desglose de todos los costes necesarios para llevar a cabo el proyecto, proporcionando un presupuesto final para el cliente.

Capítulo 7: Conclusiones y trabajos futuros

Este apartado incluye las conclusiones finales y un aporte personal acerca de la realización del proyecto. Se indicarás además posibles mejoras y funcionalidades que pueden añadirse al proyecto y una serie de proyectos que podrían desarrollarse en un futuro basándose en este.

Bibliografía

Incluye el conjunto de referencias que han sido utilizadas para el desarrollo del proyecto.

Anexo: Manual de usuario

Se trata de un manual proporcionado al usuario que detalla el funcionamiento del sistema paso a paso, de tal forma que un usuario inexperto en la materia pueda manejar el sistema de una manera sencilla.

Capítulo 2: Estado de la cuestión

Este capítulo tiene como objetivo analizar la situación del mercado actual en cuanto a las tecnologías y productos que pueden ser utilizados en este proyecto. Se analizará cada uno de ellos, teniendo en cuenta las ventajas y desventajas que pueden acarrear su uso. Finalmente se explicará cada componente, tanto hardware como software, que será utilizado en el sistema.

2.1. Sistemas de hardware libre

Se denomina hardware libre o hardware de código abierto a todo dispositivo cuyo funcionamiento, arquitectura y diseño pueden ser consultados de manera pública, de manera o no gratuita.

Son numerosos los proyectos de hardware libre disponibles para el usuario; a continuación, se exponen aquellos que tienen mayor relevancia y que pueden ser de gran utilidad para la realización de este proyecto.

Raspberry Pi

Raspberry Pi es un ordenador de bajo coste y de tamaño similar al de una tarjeta de crédito que consta de una placa base sobre la que se ensambla un procesador y una memoria RAM, entre otros componentes.

Este proyecto fue lanzado por la Fundación Raspberry Pi en el año 2009 con el objetivo de desplegarse en centros educativos para estimular el aprendizaje y desarrollar nuevas aplicaciones.

Son numerosos los proyectos que pueden realizarse con Raspberry Pi, desde un nivel básico para iniciación hasta proyectos más complejos. Actualmente es considerado uno de los proyectos de hardware libre más conocidos por el público.

Ilustración 1: Raspberry Pi

Arduino

Arduino, junto a Raspberry Pi, es uno de los proyectos de hardware libre más populares y utilizados por los usuarios.

Se trata de una placa de tamaño reducido que contiene un microcontrolador Atmel AVR y que ofrece varios puertos de entrada y salida.

La primera placa de Arduino fue introducida en el año 2005, y ofrecía, al igual que actualmente, un bajo coste, con el fin de fomentar el uso y desarrollar proyectos educativos. En la actualidad existen varios modelos de placa Arduino, para poder realizar desde proyectos básicos educativos hasta sistemas más complejos.

Es la base de proyectos relacionados con la electrónica, y cada vez se está utilizando más para proyectos de domótica. Posee un entorno de desarrollo propio para la creación de aplicaciones, gracias a un lenguaje también propio, Arduino, similar a otros lenguajes como Java o C.

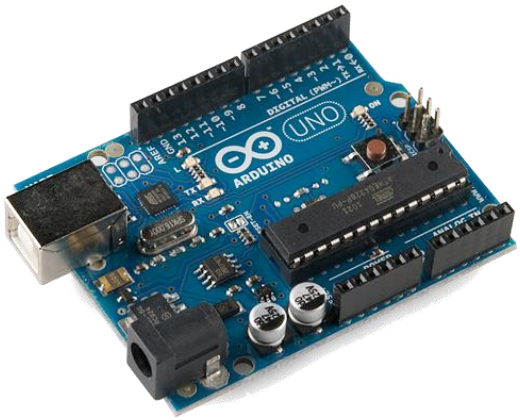


Ilustración 2: Arduino

Uzebox

Uzebox consiste en un proyecto de hardware libre cuyo objetivo es el desarrollo de una consola con las características similares a los videojuegos clásicos de 16 bits (cuarta generación), como pueden ser el Tetris o el Pac-Man. Permite además conectar un mando de la consola NES o de la Super Nintendo para su uso.

La consola Uzebox es una placa formada por un microcontrolador AVR de 8 bits de Amtel y una memoria RAM de 4KB y 64KB de programa. Pueden llegar a desplegarse hasta 256 colores, y el kernel sincroniza el audio y el vídeo a tiempo real.

Existen reposiciones de más de 50 videojuegos clásicos de la década de los 80 disponibles para su desarrollo en Uzebox.

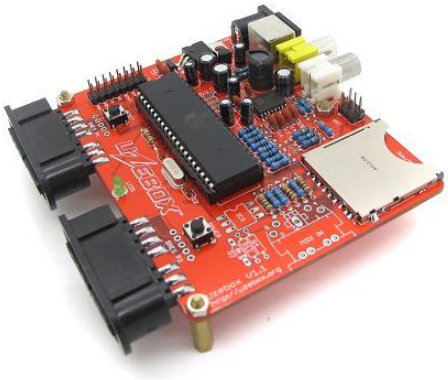


Ilustración 3: Uzebox

Cubieboard

Este proyecto de hardware libre consiste en un ordenador de placa reducida similar a Raspberry Pi pero con unas mejores prestaciones en cuanto a rendimiento. Actualmente puede afirmarse que es el potencial competidor de Raspberry Pi.

Ofrece 1 GB de memoria RAM y un procesador ARM A10 de 1GHz. Posee un almacenamiento de 4GB, y puede conectarse un disco duro SATA.

Por defecto en la placa se encuentra instalado Android 4.0.4, aunque por su procesador y memoria, se puede acceder e instalar distribuciones más potentes como Ubuntu y aplicaciones que requieran un mayor rendimiento.

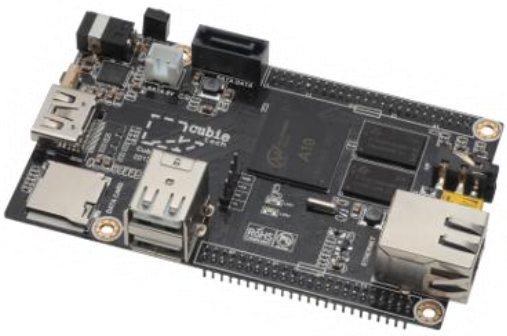


Ilustración 4: Cubieboard

RepRap Project

Es un proyecto iniciado en el año 2004, que combina la filosofía de hardware libre con la impresión en 3D. Se trata de una impresora básica en 3D que cuenta con una característica especial, que es la capacidad de autorreplicarse.

Con una impresora RepRap, mediante la impresión de sus componentes, se puede llegar a replicar la misma impresora. En la actualidad sigue siendo un proyecto en continuo desarrollo, ya que puede ser útil para facilitar que la impresión en 3D pueda llegar a cualquier persona en un futuro.

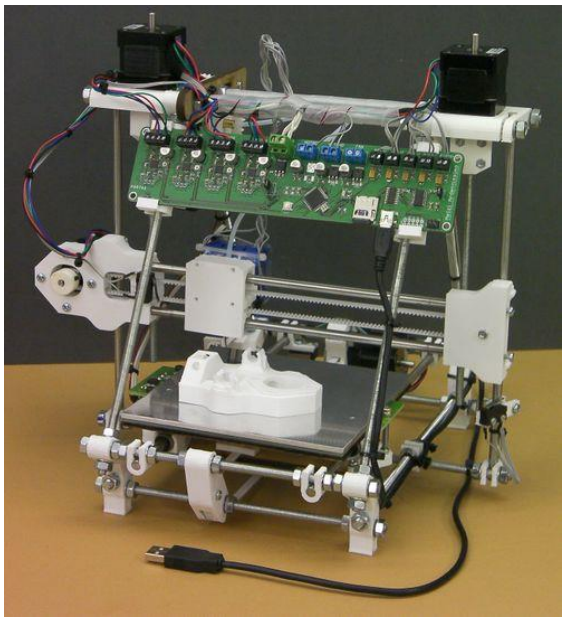


Ilustración 5: Impresora RepRap

2.2. Tecnologías móviles

En el proyecto se realizará una aplicación para teléfono móvil con la finalidad de controlar el vehículo, por lo que será necesario realizar un análisis de las tecnologías móviles más utilizadas en la actualidad.

Android

Se trata de un sistema operativo diseñado para dispositivos móviles de pantalla táctil, como smartphones o tablets, adquirido por la empresa Google desde el año 2005, y basado en el núcleo Linux y desarrollado en lenguaje C.

El primer teléfono móvil con Android fue el HTC Dream, vendido en octubre de 2008, por un coste estimado de 180 dólares.

Una de las principales ventajas del sistema operativo Android es que es un sistema operativo de código abierto y la mayoría de aplicaciones son gratuitas. Además, ofrece diferentes programas que no requieren licencia para desarrollar aplicaciones, como Android Studio.

Cualquier aplicación puede ser subida a la tienda oficial de descargas de Android, Play Store, abonado una cuota correspondiente; por este motivo, es un sistema operativo que permite la instalación y ejecución de aplicaciones desarrolladas por terceros, lo que puede suponer una ventaja, al ofrecer un número mayor de aplicaciones gratuitas y una mayor variedad, y una desventaja al poder instalar aplicaciones inseguras.

En la actualidad, la cuota de mercado de dispositivos inteligentes con Android supera la de otros sistemas operativos, como son iOS y Windows Phone juntos. En España, recientemente (junio de 2016), se ha superado el número de ventas de teléfonos con Android, un porcentaje estimado de 9 de cada 10 dispositivos.

iOS

iOS es el sistema operativo para móviles de la multinacional Apple. Originalmente fue desarrollado para iPhone, pero en la actualidad se encuentra en otros dispositivos, como el iPad.

A diferencia de otros sistemas operativos, este no permite la instalación de software de terceros, sino que únicamente se

permite software original de Apple.

Las aplicaciones para iOS son descargadas desde la tienda oficial, Apple Store, donde el número de aplicaciones disponibles es menor que para Android.

Actualmente, la cuota de mercado de dispositivos iOS muestra un crecimiento en ventas desde inicios de 2016, pero sigue siendo Android el sistema operativo más vendido en el mundo.

Windows Phone

Es el sistema operativo para móviles desarrollado por Microsoft, presentado en el mercado en el año 2010 como sucesor de Windows Mobile.

A diferencia de este último, Windows Phone está enfocado a un entorno de mercado más comercial y cercano al usuario; posee una interfaz gráfica similar a la del sistema operativo para ordenadores Windows 8, e integra varios de sus servicios propios, como son Skype, Dropbox o OneDrive.

Es el tercer competidor potencial en la venta de teléfonos móviles, junto a iOS y Android, pero en comparación con estos dos, el número de ventas es muy inferior, en el caso de España ocupa una cuota de mercado de cerca del 2%.

La tienda oficial de aplicaciones es una plataforma web, donde los usuarios pueden descargar aplicaciones, música o juegos, entre otros. Cuenta con un número de aplicaciones total de 560.000, inferior al número de aplicaciones que poseen las tiendas oficiales de sus dos potenciales competidores.

2.3. Hardware utilizado para la implementación del sistema

2.3.1. Arduino UNO

Para la realización de este proyecto se utilizará Arduino UNO; se trata del modelo de placa Arduino más básico y es el que se utiliza habitualmente para proyectos de esta magnitud. Utiliza un microcontrolador Atmega en formato DIP.

La función de Arduino será recibir y procesar las órdenes que el usuario envíe desde el teléfono móvil para controlar el vehículo.

La siguiente tabla muestra las especificaciones técnicas más relevantes de la placa Arduino UNO utilizada:

Características técnicas Arduino UNO	
Microcontrolador	Atmega328
Voltaje de operación	5V
Voltaje de entrada recomendado	7 – 12V
Límites voltaje de entrada	6 – 20V
Pines para entrada-salida digital	14 (6 pueden usarse como salida PWM)
Pines de entrada analógica	6
Corriente continua por pin IO	40 mA
Corriente continua en pin 3.3V	50 mA
Memoria flash	32 KB (0.5 KB para el arranque)
SRAM	2 KB
EEPROM	1 KB
Frecuencia de reloj	16 MHz

Tabla 1: Características técnicas Arduino UNO

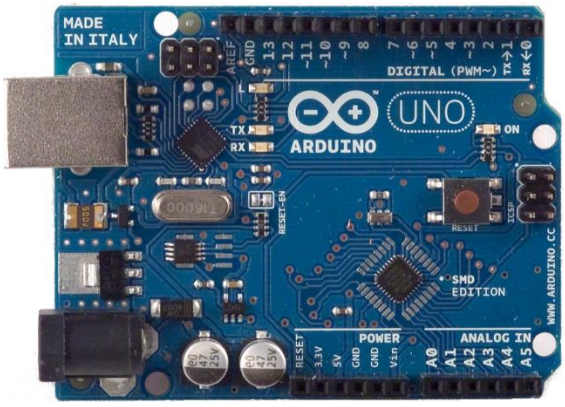


Ilustración 6: Placa Arduino UNO

2.3.2. Chip L293D

Este chip servirá para generar movimiento en los dos motores de corriente continua que poseerá el vehículo; para cada motor se realizará un puente H a través de este chip, y así se podrán controlar los posibles giros del motor: girar hacia adelante y girar hacia atrás.

La implementación y lógica del puente H, así como la funcionalidad interna de este chip, se explicará de manera detallada en el capítulo 4 de este documento.

Características técnicas L293D	
Capacidad de corriente de salida por canal	600 mA
Corriente máxima	1 A
Voltajes de operación	4.5 – 36 V
Número de pines	16

Tabla 2: Características técnicas L293D

Ilustración 7: Chip L293D

2.3.3. Módulo Bluetooth HC-06

Esta componente es utilizada para realizar la conexión entre el teléfono móvil y el Arduino vía Bluetooth, y así poder transferir los datos que el usuario envía.

En la siguiente tabla se muestran las características técnicas más relevantes del módulo Bluetooth HC-06:

Características técnicas HC-06	
Protocolo Bluetooth	2.0 + EDR (Enhanced Data Rate)
Modo	Eslavo (sólo puede operar en este modo)
Frecuencia	2.4 GHz
Alcance	5 a 10 metros
Voltaje de operación	3.6 – 6 voltios
Temperatura de operación	-25ºC – 75ºC
Seguridad	Autenticación y encriptación (contraseña por defecto: 1234)
Tasa de transferencia (por defecto)	9600 baudios
Consumo de corriente	30 mA – 40 mA
Número de pines	4

Tabla 3: Características técnicas HC-06

Ilustración 8: Módulo HC-06

2.3.4. Sensor ultrasónico HC-SR04

Se trata de un dispositivo cuya finalidad es medir la distancia a la que se encuentra un objeto situado frontalmente, mediante un par de transductores de ultrasonido.

Este sensor formará parte del sistema anti-colisiones que tendrá el vehículo, ya que será capaz de detectar obstáculos a una cierta distancia. El funcionamiento tanto del sensor como del software que requiere, será detallado en el capítulo 4 de este documento.

La siguiente tabla recoge las características técnicas de este sensor de ultrasonidos:

Características técnicas HC-SR04	
Tensión de alimentación	5 V
Frecuencia de trabajo	40 KHz
Rango de distancias	1.7 centímetros – 4.5 metros
Alcance	5 a 10 metros
Corriente de trabajo	15 mA
Ángulo de medición	30º (efectivo <15º)
Número de pines	4

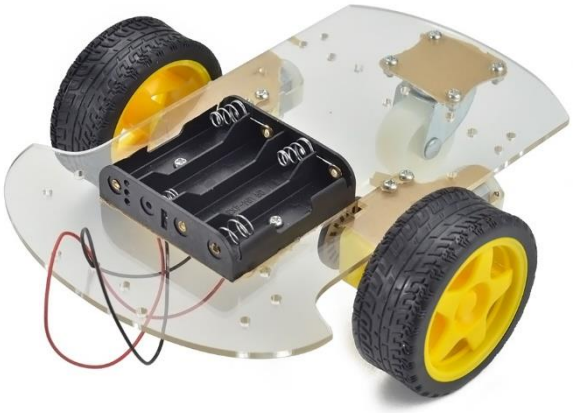
Tabla 4: Características técnicas HC-SR04

Ilustración 9: Sensor ultrasónico HC-SR04

2.3.5. Estructura del vehículo

El vehículo que se utilizará para este proyecto está formado por diversas componentes:

- **Tabla de metacrilato**: es una tabla sobre la que se posará la circuitería del sistema: placa Arduino, conexiones con la placa de pruebas, baterías, sensores…
- **Ruedas**: el vehículo constará de dos ruedas con tracción, conectadas a dos motores eléctricos, y de una rueda loca, que es una rueda que gira en todas direcciones sin tener tracción.
- **Motores**: se utilizarán dos motores eléctricos para generar giro en las ruedas.



2.3.6. Protoboard o placa de pruebas

Se ha utilizado, además, una placa de pruebas de 300 para que pueda adaptarse en la estructura del vehículo.

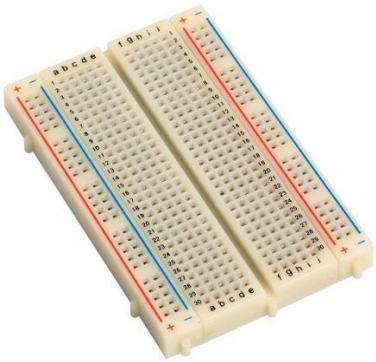


Ilustración 10: Placa de pruebas

2.4. Software utilizado para la implementación del sistema

2.4.1. Android Studio

Android Studio es un entorno de desarrollo de aplicaciones para dispositivos con sistema operativo Android. Este software puede descargarse gratuitamente desde su página oficial.

Las aplicaciones se desarrollan mediante código Java, lo que supone una mayor complejidad comparado con la aplicación desarrollada en el proyecto. Esta aplicación también podría desarrollarse con Android Studio, y tendría la ventaja de adquirir una mayor flexibilidad y escalabilidad, pero al tratarse de una aplicación tan sencilla se tardaría más tiempo en programarla con código Java.

Otra de las ventajas que ofrece este software, aparte de permitir crear aplicaciones más complejas, es la pre-visualización de la aplicación en desarrollo en un emulador, descargable a través del propio software. Este emulador simula un dispositivo Android, pudiendo elegir entre distintos tamaños de pantalla y distintas versiones de Android. La aplicación es instalada en el emulador, y el usuario puede acceder a su funcionalidad como si fuera un teléfono móvil real.

Ilustración 11: Android Studio

La desventaja del software es que consume mucha memoria RAM; los desarrolladores recomiendan mínimo 2GB, pero en aplicaciones que tengan mucho código fuente, el consumo puede ser superior. El software, a pesar de no ocupar demasiado espacio en disco duro, 400 MB aproximadamente, requiere la instalación de Android SDK, que es 1GB, y los emuladores que se instalen para realizar las pruebas ocupan cada uno mínimo 1GB.

El ordenador en el que se ha desarrollado la aplicación cumple con los requisitos de memoria RAM y procesador, ya que posee 8 GB de RAM y un Intel Core i7, por lo que no es una limitación desarrollar la aplicación con este software.

Pese a que existen otras herramientas de desarrollo de aplicaciones Android de un uso más sencillo, y pese a que la aplicación que se va a desarrollar es sencilla, se ha decidido utilizar Android Studio ya que incluye más flexibilidad en cuanto a funcionalidades y en cuanto a diseño de la interfaz de usuario.

2.4.2. Arduino IDE

Es el entorno de programación usado para el desarrollo del código que ejecutará el Arduino. Se puede descargar de manera gratuita desde la página oficial de Arduino.

El lenguaje Arduino está basado en C, por lo que es necesario tener conocimientos, al menos básicos, de C o C++. Todas las librerías que utiliza Arduino pueden consultarse en la página web o en el manual que se ofrece de forma libre y gratuita.

En la siguiente imagen se puede ver la estructura básica con la que se encuentra el usuario al empezar a programar una aplicación; en ella aparecen dos funciones, setup() y loop().

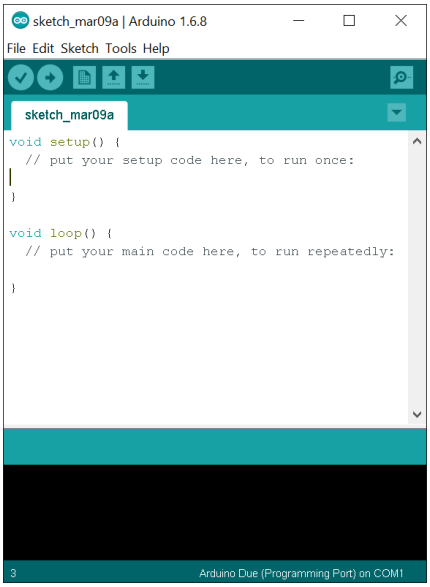


Ilustración 12: Programa Arduino IDE

La función setup() es la encargada de inicializar las configuraciones de los distintos elementos del programa, como puede ser configurar un pin determinado de entrada o salida, o establecer la velocidad de comunicación por el puerto serial. Esta función la ejecuta el Arduino una sola vez al iniciar el programa; para que vuelva a ejecutarse, el Arduino se debe apagar o resetear.

La función loop() contiene todas las instrucciones del programa, que serán ejecutadas un número indeterminado de veces, como el propio nombre de la función indica. Al llegar al final de la función loop(), se vuelve a empezar la función con los cambios que hayan sido realizados a lo largo del programa.

En un programa de Arduino se ejecuta el código línea a línea, es decir, primero se ejecuta la función setup() y después continua con la función loop().

2.5. Alternativas al hardware utilizado

Para la realización de este proyecto, se puede utilizar otro modelo de placa Arduino, ya que el software de desarrollo es válido para cualquier modelo y el funcionamiento no se hubiera visto modificado.

Arduino Mega es una alternativa cuyo uso se calibró al inicio de este proyecto, pero finalmente se decidió utilizar Arduino UNO por motivos de coste y de prestaciones, ya que, para un proyecto sencillo, las dos placas ofrecerían las mismas prestaciones y el coste de Arduino UNO desciende cerca de la mitad.

2.5.1. Arduino Mega

Arduino Mega es otro de los modelos de placa Arduino que se pueden utilizar para realizar este proyecto. Cuenta con mejores prestaciones que Arduino UNO, como un mayor número de pines digitales y analógicos, un microcontrolador más potente o más memoria, y es apto para proyectos más complejos que involucren un mayor número de conexiones. También optimiza el consumo de corriente continua de entrada/salida, la mitad que Arduino UNO.

Como se comentó anteriormente, para este proyecto es suficiente utilizar Arduino UNO, ya que Arduino Mega proporcionaría las mismas prestaciones, y al ser más complejo, su precio incrementa cerca del doble.

La siguiente tabla ofrece una comparativa de las prestaciones de Arduino UNO y Mega:

	Arduino UNO	Arduino Mega
Microcontrolador	ATmega328	ATmega2560
Voltaje de operación	5V	5V
Voltaje de entrada recomendado	7 – 12V	7 – 12V
Límites voltaje de entrada	6 – 20V	6 – 20V
Pines para entrada-salida digital	14 (6 PWM)	54 (15 PWM)
Pines de entrada analógica	6	16
Corriente continua por pin IO	40 mA	20 mA
Corriente continua en pin 3.3V	50 mA	50 mA
Memoria flash	32 KB (0.5 KB arranque)	256 KB (8KB arranque)
SRAM	2 KB	8 KB
EEPROM	1 KB	4 KB
Frecuencia de reloj	16 MHz	16 MHz
Precio	25€	45€

Tabla 5: Comparativa Arduino UNO vs Mega

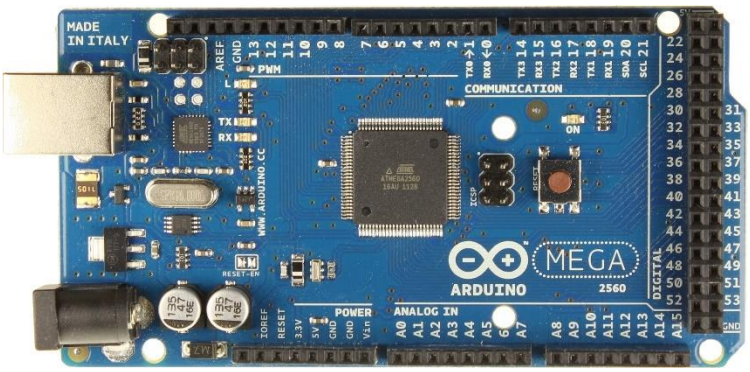


Ilustración 13: Placa Arduino Mega

2.5.2. Mando de infrarrojos

Otra forma de manejar Arduino de forma remota es mediante el uso de un mando de infrarrojos; el funcionamiento es el mismo que el de cualquier mando a distancia, por ejemplo, el de un televisor o el de un aire acondicionado.

El mando emite una señal infrarroja que llega hasta un receptor de infrarrojos, como puede ser el AX-1838HS, y es recibida por el Arduino, que la procesa y realiza la acción que se desee. El ejemplo más típico es encender un led con el mando, aunque tiene otras utilidades.

Ilustración 14: Mando de infrarrojos

Para la realización de este proyecto se podría haber utilizado un mando de infrarrojos para controlar de forma remota el vehículo; bastaría con modificar el circuito, empezando por sustituir el receptor Bluetooth por el receptor de infrarrojos, y programar el software para que, dependiendo de la señal emitida, realizar una acción en el vehículo u otra.

Esta tecnología no se ha implementado en este proyecto dado que la idea principal es manejar el vehículo desde un teléfono móvil y no desde un mando. La comunicación entre el vehículo y el teléfono también se hubiera podido realizar mediante infrarrojos, pero se descartó enseguida dado que es una tecnología que se encuentra en desuso, y el Bluetooth permite más flexibilidad y tiene más ventajas.

2.6. Alternativas al software utilizado

2.6.1. MIT App Inventor

Se trata de una plataforma de Google que permite diseñar y crear aplicaciones para el sistema operativo Android de manera sencilla y visual.

La herramienta está constituida por dos pantallas principales: la primera es el diseñador, en la que se permite al usuario diseñar la interfaz de usuario de la aplicación, por ejemplo, añadir un número determinado de pantallas, añadir botones, imágenes, etc. También puede incluir otros aspectos, como conectividades de Bluetooth, elementos multimedia o bases de datos para almacenamiento en el teléfono.

La siguiente ilustración corresponde al diseñador de la aplicación; en la parte central se puede ver una pantalla con la interfaz de usuario de la aplicación, donde el usuario añade botones, imágenes o cajas de texto, entre otras opciones, que pueden encontrarse en el menú de la izquierda.

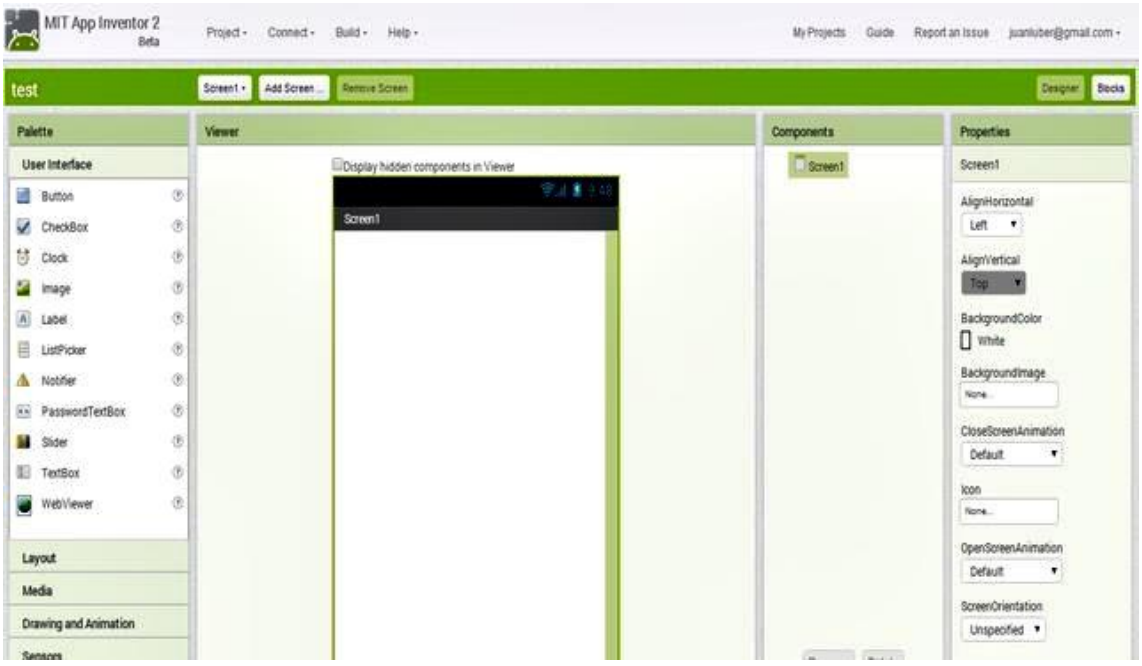


Ilustración 15: MIT App Inventor 2

La segunda pantalla es la que permite realizar la parte lógica y la funcionalidad de la aplicación. Esto se realiza de manera visual e intuitiva, a través de una serie de bloques y cajas que indican una determinada funcionalidad o condición.

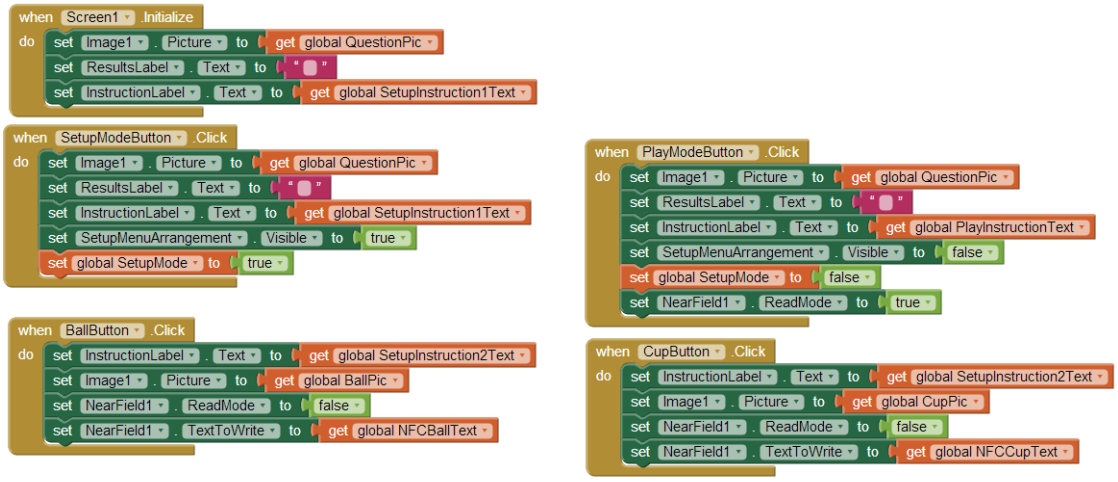


Ilustración 16: MIT App Inventor 2

La principal ventaja de utilizar esta herramienta es la sencillez de uso, dado que ofrece al usuario tanto la posibilidad de diseñar la interfaz de la aplicación como implementar la lógica de una manera muy sencilla e intuitiva. A pesar de que no se requiera programar como tal, el usuario debe tener nociones básicas de programación y lógica.

Esta aplicación también tiene una serie de desventajas y limitaciones; la principal desventaja frente a desarrollar programando en código Java es que es menos flexible, y por lo tanto permite implementar un número inferior de funcionalidades frente a otros entornos. Esto se traduce en que utilizar este programa es válido únicamente para aplicaciones sencillas que no requieran una funcionalidad demasiado compleja.

Las limitaciones que puede tener esta aplicación son, que para utilizarla es necesario disponer de una conexión a Internet, al ser una herramienta online, y también es necesario disponer de una cuenta de Gmail al ser desarrollada por Google.

2.6.2. Eclipse

Eclipse es una plataforma de desarrollo de software, compuesta por una serie de diferentes proyectos, y apta para desarrollar en distintos lenguajes como Java, C++ o JSP.

En las últimas versiones de Eclipse también se puede desarrollar aplicaciones para Android; para ello es necesario descargar, aparte del kit de desarrollo de Java (JDK), un plugin gratuito que contiene todas las herramientas de desarrollo de Android.

Esta es la principal desventaja que tiene eclipse frente a Android Studio, y el motivo por el que este último está ganando terreno a la hora de desarrollar aplicaciones Android, ya que permite al usuario programar directamente sin tener que instalar previamente las herramientas necesarias.

Por ello, en el caso de futuros proyectos o mejoras de la funcionalidad de este en cuanto a la aplicación, se continuaría utilizando la herramienta Android Studio.

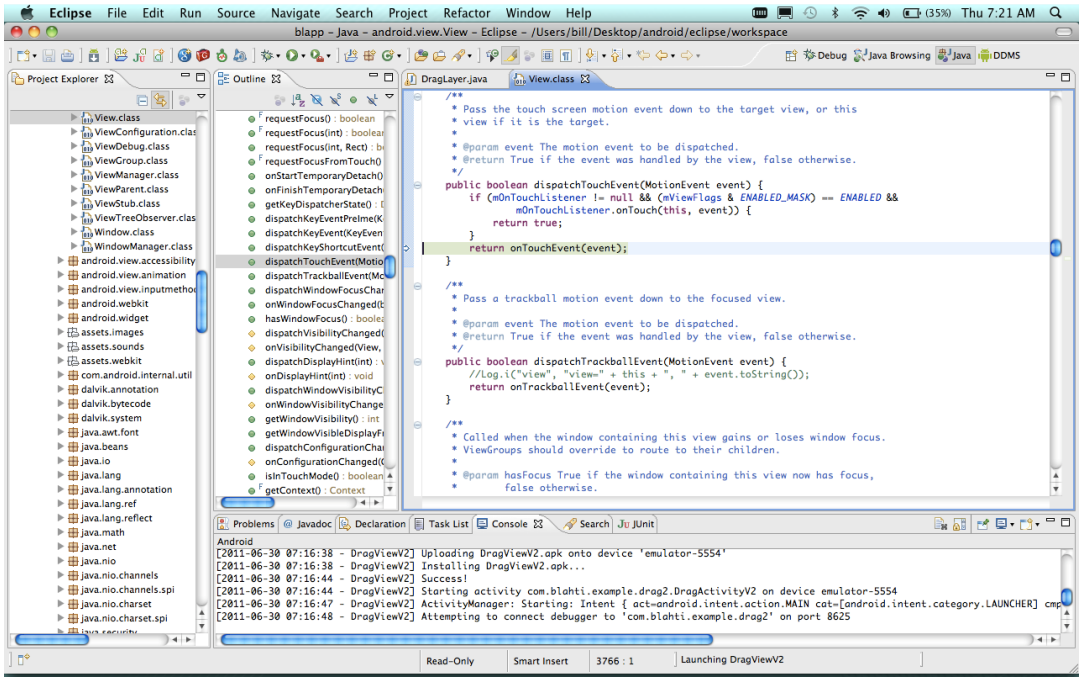


Ilustración 17: Software Eclipse

2.6.3. Miniblog

Es una herramienta de código abierto cuyo objetivo es el desarrollo de aplicaciones para Arduino, proporcionando al usuario un entorno de desarrollo gráfico, y diseñado principalmente para la iniciación a la programación de Arduino.

La aplicación contiene una serie de bloques que proporcionan distintas funcionalidades; el usuario, mediante la unión de dichos bloques, desarrolla la aplicación, cuyo funcionamiento puede ser probado gracias al terminal interactivo que proporciona. Además, la herramienta incluye distintos tipos de placas de Arduino.

La ventaja es la sencillez de uso para cualquier usuario que se inicie en el desarrollo de aplicaciones para Arduino, pero esta herramienta, en comparación con otras más complejas como la que se ha utilizado para este proyecto, presenta numerosas limitaciones de funcionalidad, por lo que aplicaciones que requieran mayor complejidad no pueden ser desarrolladas con Minibloq.

Un ejemplo de la interfaz de la aplicación se muestra en la siguiente imagen:

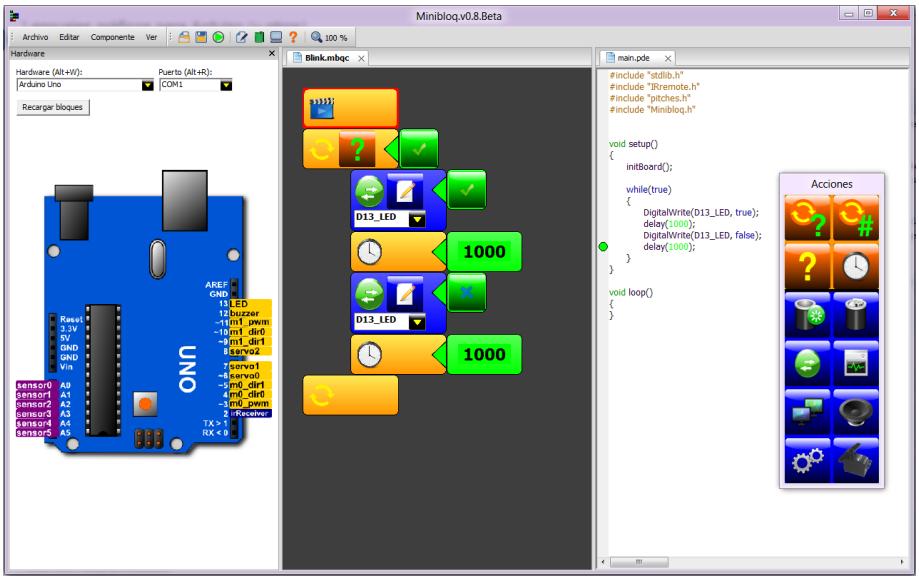


Ilustración 18: Minibloq

2.6.4. Modkit

Modkit es otra herramienta de desarrollo de aplicaciones para Arduino que presenta un entorno gráfico al usuario. El funcionamiento es similar al de la herramienta Minibloq, ya que las aplicaciones son desarrolladas también gracias a la unión de bloques que proporcionan las funcionalidades posibles y las traducen a código Arduino.

Esta herramienta también sirve de iniciación a la programación de Arduino y es válida para el desarrollo de aplicaciones sencillas, pero para un proyecto de mayor magnitud presenta bastantes limitaciones de uso.

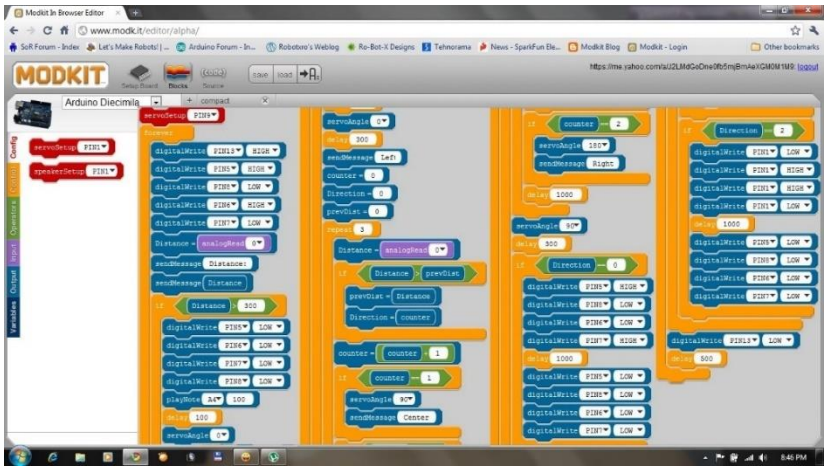


Ilustración 19: Modkit

Capítulo 3: Análisis del sistema

El propósito de este capítulo es la especificación detallada de los requisitos que se tendrán en cuenta para el posterior diseño e implementación del sistema. Se diferenciarán entre requisitos de usuario y requisitos de sistema. Además, se detallarán todos los escenarios posibles de los casos de uso del sistema.

Para definir los requisitos del sistema, estos se clasificarán en dos categorías principales:

- **Requisitos de usuario**: se trata de los requisitos acordados con el cliente para describir la funcionalidad del sistema. Se deben expresar en un lenguaje natural, que el cliente pueda entender, sin vocabulario técnico y sin entrar en detalles. Pueden ser:
    - **Requisitos de capacidad**: describen una funcionalidad del sistema.
    - **Requisitos de restricción**: describen restricciones que deben tenerse en cuenta en el diseño.

- **Requisitos de sistema**: los requisitos del sistema son una descripción en un lenguaje técnico de los requisitos de usuario. Para cada requisito de usuario, al menos debe existir un requisito de sistema. Pueden ser:
    - **Requisitos funcionales**: hace referencia a un requisito de usuario de capacidad.
    - **Requisitos no funcionales**: hace referencia a un requisito de usuario de restricción.

Una vez establecidos todos los requisitos de usuario y de sistema, se establecerá una matriz de trazabilidad entre estos para comprobar la validación entre requisitos, y verificar así que cada requisito de sistema hace referencia, al menos, a un requisito de usuario.

Los requisitos se especificarán utilizando el siguiente formato de tabla:

Identificador: R(N)-(X)-(YY)	
Descripción:	
Necesidad:	
Claridad:	
Prioridad:	
Estabilidad:	
Verificabilidad:	
Origen:	

Tabla 6: Formato tabla requisitos

Los campos que componen la tabla son:

- **Identificador**: código que identificará cada requisito de forma unívoca para facilitar la trazabilidad. Este código seguirá la siguiente nomenclatura:
    - **R**: correspondiente a la sigla de "Requisito". Esta letra será común tanto para requisitos de usuario como para requisitos de sistema.
    - **N**: este segundo carácter podrá tomar los siguientes valores:
        - **U**: si se trata de un requisito de usuario.
        - **S**: si se trata de un requisito de sistema.
    - **XX**: estos dos caracteres pueden ser:
        - Si se trata de un requisito de usuario
            - **C**: es un requisito de **capacidad**.
            - **R**: es un requisito de **restricción**.
        - Si se trata de un requisito de sistema
            - **F**: es un requisito **funcional**.
            - **NF**: es un requisito **no funcional**.
    - **YY**: número de requisito dentro de la categoría principal correspondiente. Será un número compuesto de dos cifras que pueden tomar valores entre 01 y 99.

- **Descripción**: campo con la descripción en detalle del requisito.

- **Necesidad**: indica el grado de necesidad de incorporación del requisito en el sistema. Este campo tomará únicamente uno de los siguientes valores:
    - **ESENCIAL**: la incorporación de este requisito en el sistema es obligatoria.
    - **DESEABLE**: se estima la incorporación de este requisito en el sistema, aunque está sujeta a negociaciones.
    - **OPCIONAL**: la incorporación de este requisito en el sistema no es obligatoria.

- **Claridad**: indica si el requisito está descrito y expresado de una forma correcta, de tal forma que se evite la ambigüedad. Este campo tomará uno de los siguientes valores:
    - **ALTA**: el requisito tiene solamente una interpretación.
    - **MEDIA**: el requisito está bien definido.
    - **BAJA**: el requisito puede ser interpretado de varias formas debido a la ambigüedad de su descripción y debe ser modificado para conseguir una mejor claridad.

- **Prioridad**: indica el grado de prioridad del requisito. Este campo tomará uno de los siguientes valores:
    - **ALTA**: requisito que se debe implementar con alto grado de prioridad.

    - **MEDIA**: requisito que se debe implementar con un grado de prioridad medio.
    - **BAJA**: requisito que se debe implementar con una prioridad baja.

- **Estabilidad**: indica el grado de posibilidad de que el requisito se modifique durante el desarrollo del proyecto. Este campo tomará uno de los siguientes valores:
    - **ALTA**: es poco probable que el requisito sea modificado durante el desarrollo del proyecto.
    - **MEDIA**: la probabilidad de que el requisito sufra modificaciones durante el desarrollo del proyecto es media.
    - **BAJA**: es muy probable que el requisito sea modificado durante el desarrollo del proyecto.

- **Verificabilidad**: indica el grado en el que es posible comprobar la inclusión del requisito en el sistema. Este campo tomará uno de los siguientes valores:
    - **ALTA**: la inclusión del requisito en el sistema se puede comprobar de forma sencilla.
    - **MEDIA**: la inclusión del requisito en el sistema se puede comprobar sin una dificultad elevada o relevante.
    - **BAJA**: la inclusión del requisito en el sistema es difícil de comprobar.

- **Origen**: indica la fuente origen del requisito. Este campo podrá hacer referencia a:
    - **CLIENTE**: si el cliente del proyecto ha impuesto el requisito. Aparecerá en los requisitos de usuario principalmente.
    - **REQUISITO**: toma como referencia un requisito. Esto se da, fundamentalmente, en los requisitos de sistema, ya que hacen referencia a un requisito de usuario.

3.1. Requisitos de usuario

3.1.1. Requisitos de capacidad

RU-C-01	
Descripción:	El usuario debe tener una conexión inalámbrica habilitada en el dispositivo móvil.
Necesidad:	Esencial
Claridad:	Alta
Prioridad:	Alta
Estabilidad:	Baja
Origen	Cliente
Verificabilidad:	Alta

Tabla 7: Requisito de usuario RU-C-01

RU-C-02	
Descripción:	Se debe poder seleccionar el dispositivo con el que establecer la conexión inalámbrica.
Necesidad:	Esencial
Claridad:	Alta
Prioridad:	Alta
Estabilidad:	Baja
Origen	Cliente
Verificabilidad:	Alta

Tabla 8: Requisito de usuario RU-C-02

RU-C-03	
Descripción:	Debe existir una conexión entre el vehículo y el dispositivo móvil.
Necesidad:	Esencial
Claridad:	Alta
Prioridad:	Alta
Estabilidad:	Baja
Origen	Cliente
Verificabilidad:	Alta

Tabla 9: Requisito de usuario RU-C-03

RU-C-04	
Descripción:	El tiempo de conexión entre dispositivos debe ser inferior a 15 segundos.
Necesidad:	Esencial
Claridad:	Alta
Prioridad:	Alta
Estabilidad:	Baja
Origen	Cliente
Verificabilidad:	Alta

Tabla 10:Requisito de usuario RU-C-04

RU-C-05	
Descripción:	Al iniciar la aplicación se presentará al usuario el modo manual por defecto.
Necesidad:	Esencial
Claridad:	Alta
Prioridad:	Alta
Estabilidad:	Baja
Origen	Cliente
Verificabilidad:	Alta

Tabla 11: Requisito de usuario RU-C-05

RU-C-06	
Descripción:	El usuario debe poder seleccionar entre modo de conducción del vehículo automático y manual.
Necesidad:	Esencial
Claridad:	Alta
Prioridad:	Alta
Estabilidad:	Baja
Origen	Cliente
Verificabilidad:	Alta

Tabla 12: Requisito de usuario RU-C-06

RU-C-07	
Descripción:	El usuario debe poder manejar el vehículo en modo manual (mover hacia adelante, hacia atrás, girar a izquierda y a derecha y detener).
Necesidad:	Esencial
Claridad:	Alta
Prioridad:	Alta
Estabilidad:	Baja
Origen	Cliente
Verificabilidad:	Alta

Tabla 13: Requisito de usuario RU-C-07

RU-C-08	
Descripción:	El usuario no podrá manejar el vehículo en modo automático.
Necesidad:	Esencial
Claridad:	Alta
Prioridad:	Alta
Estabilidad:	Baja
Origen	Cliente
Verificabilidad:	Alta

Tabla 14: Requisito de usuario RU-C-08

RU-C-09	
Descripción:	El diseño de la interfaz de usuario de la aplicación debe ser sencillo e intuitivo.
Necesidad:	Esencial
Claridad:	Alta
Prioridad:	Alta
Estabilidad:	Baja
Origen	Cliente
Verificabilidad:	Alta

Tabla 15: Requisito de usuario RU-C-09

RU-C-10	
Descripción:	Se debe poder desconectar el dispositivo con el que se estableció la conexión inicial.
Necesidad:	Esencial
Claridad:	Alta
Prioridad:	Alta
Estabilidad:	Baja
Origen	Cliente
Verificabilidad:	Alta

Tabla 16: Requisito de usuario RU-C-10

RU-C-11	
Descripción:	La aplicación deberá notificar al usuario si se produce cualquier tipo de error.
Necesidad:	Esencial
Claridad:	Alta
Prioridad:	Alta
Estabilidad:	Baja
Origen	Cliente
Verificabilidad:	Alta

Tabla 17: Requisito de usuario RU-C-11

RU-C-12	
Descripción:	El tiempo de respuesta de la aplicación debe ser inferior a 30 segundos.
Necesidad:	Esencial
Claridad:	Alta
Prioridad:	Alta
Estabilidad:	Baja
Origen	Cliente
Verificabilidad:	Alta

Tabla 18: Requisito de usuario RU-C-12

RU-C-13	
Descripción:	El vehículo deberá disponer de un dispositivo para realizar conexiones inalámbricas.
Necesidad:	Esencial
Claridad:	Alta
Prioridad:	Alta
Estabilidad:	Baja
Origen	Cliente
Verificabilidad:	Alta

Tabla 19: Requisito de usuario RU-C-13

RU-C-14	
Descripción:	El dispositivo inalámbrico del vehículo debe estar encendido y visible al usuario.
Necesidad:	Esencial
Claridad:	Alta
Prioridad:	Alta
Estabilidad:	Baja
Origen	Cliente
Verificabilidad:	Alta

Tabla 20: Requisito de usuario RU-C-14

RU-C-15	
Descripción:	El vehículo deberá estar dotado de un sistema anti-colisiones para evitar choques frontales.
Necesidad:	Esencial
Claridad:	Alta
Prioridad:	Alta
Estabilidad:	Baja
Origen	Cliente
Verificabilidad:	Alta

Tabla 21:Requisito de usuario RU-C-15

RU-C-16	
Descripción:	El vehículo deberá poseer una fuente de alimentación propia.
Necesidad:	Esencial
Claridad:	Alta
Prioridad:	Alta
Estabilidad:	Baja
Origen	Cliente
Verificabilidad:	Alta

Tabla 22: Requisito de usuario RU-C-16

RU-C-17	
Descripción:	El vehículo deberá poseer un mecanismo para generar tracción en dos ruedas.
Necesidad:	Esencial
Claridad:	Alta
Prioridad:	Alta
Estabilidad:	Baja
Origen	Cliente
Verificabilidad:	Alta

Tabla 23: Requisito de usuario RU-C-17

RU-C-18	
Descripción:	El software cargado en Arduino deberá recibir y procesar correctamente las órdenes enviadas desde la aplicación móvil.
Necesidad:	Esencial
Claridad:	Alta
Prioridad:	Alta
Estabilidad:	Baja
Origen	Cliente
Verificabilidad:	Alta

Tabla 24:Requisito de usuario RU-C-18

RU-C-19	
Descripción:	El Arduino deberá estar alimentado mediante una fuente de alimentación externa.
Necesidad:	Esencial
Claridad:	Alta
Prioridad:	Alta
Estabilidad:	Baja
Origen	Cliente
Verificabilidad:	Alta

Tabla 25: Requisito de usuario RU-C-19

3.1.2. Requisitos de restricción

RU-R-01	
Descripción:	La aplicación debe funcionar correctamente en la última versión de Android.
Necesidad:	Esencial
Claridad:	Alta
Prioridad:	Alta
Estabilidad:	Baja
Origen	Cliente
Verificabilidad:	Alta

Tabla 26: Requisito de usuario RU-R-01

RU-R-02	
Descripción:	La aplicación se deberá visualizar correctamente en cualquier Smartphone con una pantalla igual o inferior a 5.5 pulgadas.
Necesidad:	Esencial
Claridad:	Alta
Prioridad:	Alta
Estabilidad:	Baja
Origen	Cliente
Verificabilidad:	Alta

Tabla 27: Requisito de usuario RU-R-02

RU-R-03	
Descripción:	El lenguaje de la aplicación será el castellano.
Necesidad:	Esencial
Claridad:	Alta
Prioridad:	Alta
Estabilidad:	Baja
Origen	Cliente
Verificabilidad:	Alta

Tabla 28: Requisito de usuario RU-R-03

RU-R-04	
Descripción:	El lenguaje utilizado para la programación del software será Arduino.
Necesidad:	Esencial
Claridad:	Alta
Prioridad:	Alta
Estabilidad:	Baja
Origen	Cliente
Verificabilidad:	Alta

Tabla 29: Requisito de usuario RU-R-04

RU-R-05	
Descripción:	No se podrá utilizar más de una placa de Arduino.
Necesidad:	Esencial
Claridad:	Alta
Prioridad:	Alta
Estabilidad:	Baja
Origen	Cliente
Verificabilidad:	Alta

Tabla 30: Requisito de usuario RU-R-05

3.2. Especificación de los casos de uso

Para definir los casos de uso del sistema, una vez establecidos los requisitos de usuario, se van a dividir en dos escenarios principales:

- **Casos de uso del usuario**: se definirán los casos de uso para el control del vehículo por parte de un usuario desde la aplicación Android. Como ya se ha indicado anteriormente, el usuario puede seleccionar desde la aplicación entre modo manual y modo automático.

- **Casos de uso del vehículo**: serán los casos de uso que el vehículo puede realizar como actor principal.

Para la descripción de los casos de uso se utilizará el siguiente formato de tabla:

Identificador y Caso de uso	CU_<X>_<YY> - Caso de uso
Actores	
Objetivo	
Origen	
Precondiciones	
Escenario	
Postcondiciones	
Excepción	

Tabla 31: Formato tabla casos de uso

Los campos que contienen la tabla son:

- **Identificador**: identifica el caso de uso. Los campos que lo conforman son:
    - **CU**: es la sigla de caso de uso. Es común para todos los casos de uso independientemente del escenario.
    - **X**: corresponde al escenario sobre el que se especifica el caso de uso. Como ya se ha indicado anteriormente, puede ser:
        - U: si es un caso de uso del usuario.
        - V: si es un caso de uso del vehículo.
    - **YY**: es el número del caso de uso. Puede adquirir valores entre 01 y 99.

- **Caso de uso**: título del caso de uso.
- **Actores**: indica el o los agentes que pueden interactuar con el caso de uso.
- **Objetivo**: descripción detallada del caso de uso.

- **Precondiciones**: condiciones iniciales que deben cumplirse para realizar el caso de uso.
- **Escenario**: serie de pasos para realizar el caso de uso.
- **Postcondiciones**: estado del sistema una vez finalizado el caso de uso.
- **Excepción**: incidencias que pueden aparecer al realizar el caso de uso. La numeración indicada hace referencia al paso del escenario que puede provocar la incidencia. Para los casos donde más de uno o todos los pasos pueden provocar la misma, no se asigna ningún valor numérico. En el caso de que no se pueda producir una incidencia, se especificará como "No aplica".

3.2.1. Casos del uso de usuario

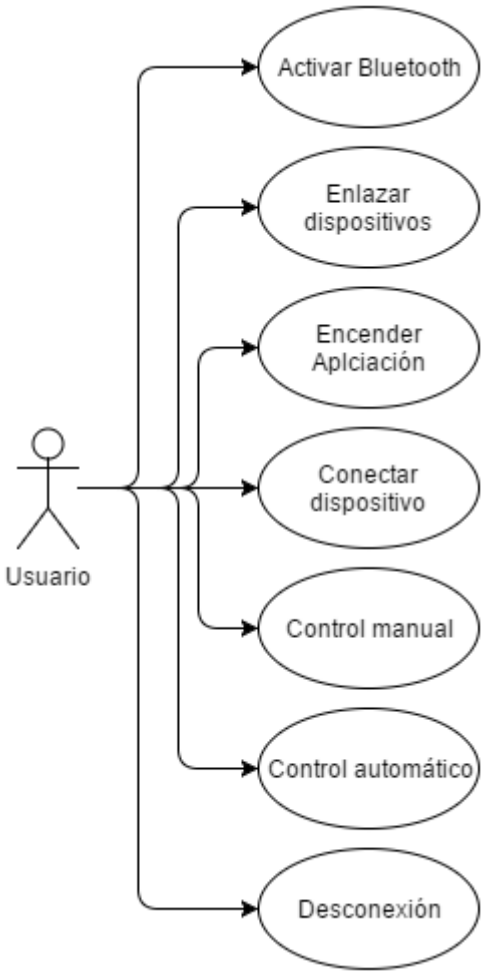


Ilustración 20: Casos de uso Usuario

Identificador y Caso de uso	CU_U_01 – Activar Bluetooth
Actores	Usuario
Objetivo	Encender el Bluetooth en el teléfono.
Precondiciones	El Bluetooth se encuentra apagado.
Escenario	1. Ir a “Conexiones”. 2. Seleccionar Bluetooth. 3. Seleccionar “Activar Bluetooth”. 4. Esperar a que el Bluetooth se active.
Postcondiciones	El Bluetooth se encuentra activado, como se muestra en la barra de estado de la pantalla del teléfono.
Excepción	La conectividad Bluetooth del teléfono no funciona.

Tabla 32: Caso de uso CU_U_01

Identificador y Caso de uso	CU_U_02 – Enlazar dispositivos
Actores	Usuario
Objetivo	Enlazar el teléfono del usuario con el dispositivo Bluetooth del vehículo para poder establecer una conexión futura.
Precondiciones	1. El dispositivo Bluetooth del vehículo y el teléfono móvil no están vinculados. 2. El dispositivo Bluetooth del vehículo se encuentra encendido.
Escenario	1. Ir a “Conexiones”. 2. Seleccionar Bluetooth. 3. Seleccionar “Activar Bluetooth”. 4. Esperar a que el Bluetooth se active. 5. Buscar dispositivos Bluetooth. 6. Seleccionar el dispositivo del vehículo. 7. Introducir contraseña. 8. Esperar a que se enlacen.
Postcondiciones	El dispositivo Bluetooth del vehículo aparece en la lista de dispositivos enlazados del teléfono.
Excepción	5. El dispositivo no se encuentra o no está disponible. 7. La contraseña es incorrecta y el dispositivo no puede enlazarse.

Tabla 33: Caso de uso CU_U_02

Identificador y Caso de uso	CU_U_03 – Encender aplicación
Actores	Usuario
Objetivo	Abrir la aplicación para poder controlar el vehículo.
Precondiciones	La aplicación debe estar instalada en el teléfono.
Escenario	1. Buscar aplicación. 2. Abrir aplicación pulsando sobre ella. 3. Esperar a que se muestre la interfaz de usuario.
Postcondiciones	Se debe visualizar correctamente la pantalla con los correspondientes botones.
Excepción	2. La aplicación se detiene fortuitamente.

Tabla 34: Caso de uso CU_U_03

Identificador y Caso de uso	CU_U_04 – Conectar dispositivo
Actores	Usuario
Objetivo	Conectarse con el dispositivo Bluetooth del vehículo.
Precondiciones	El teléfono debe estar enlazado con el dispositivo Bluetooth del vehículo.
Escenario	1. Abrir aplicación. 2. Pulsar en “Conectar dispositivo”. 3. Seleccionar de la lista de enlazados el dispositivo correspondiente al del vehículo. 4. Esperar conexión.
Postcondiciones	La aplicación notifica al usuario la correcta conexión y ambos dispositivos quedan conectados para comunicarse entre sí.
Excepción	3. El dispositivo no se encuentra en la lista. 4. Se ha producido un error al establecer la conexión.

Tabla 35: Caso de uso CU_U_04

Identificador y Caso de uso	CU_U_05 – Control manual del vehículo
Actores	Usuario
Objetivo	El usuario controla el vehículo utilizando los botones que aparecen en la interfaz de usuario de la aplicación.
Precondiciones	1. Debe existir una conexión entre el móvil y el vehículo. 2. Acabar de conectarse a un dispositivo (por defecto se inicia este modo) o estar el modo automático activado.
Escenario	1. Abrir aplicación. 2. Pulsar en “Conectar dispositivo”. 3. Seleccionar de la lista de enlazados el dispositivo correspondiente al del vehículo. 4. Esperar conexión. 5. Si la conexión es exitosa, se redirige a la pantalla de control, se activa el modo manual por defecto y los botones se habilitan.
Postcondiciones	El usuario maneja el vehículo: - Inicia marcha adelante. - Gira a izquierda y a derecha. - Detiene el vehículo. - Inicia marcha atrás.
Excepción	3. El dispositivo no se encuentra en la lista. 4. Se ha producido un error al establecer la conexión y el modo manual no se activa.

Tabla 36: Caso de uso CU_U_05

Identificador y Caso de uso	CU_U_06 – Control automático del vehículo
Actores	Usuario
Objetivo	El vehículo se mueve automáticamente hacia adelante evitando colisiones frontales.
Precondiciones	1. Debe existir una conexión entre el móvil y el vehículo. 2. Debe estar activado el modo manual.
Escenario	1. El usuario pulsa el botón de "Modo automático". 2. Se activa el modo automático.
Postcondiciones	El vehículo se mueve de forma automática hacia adelante y girando a izquierda o a derecha para evitar colisionarse con algún obstáculo que encuentre frontalmente.
Excepción	No aplica.

Tabla 37: Caso de uso CU_U_06

Identificador y Caso de uso	CU_U_07 – Desconexión
Actores	Usuario
Objetivo	El usuario sale de la aplicación.
Precondiciones	La aplicación debe estar iniciada.
Escenario	1. El usuario presiona el botón de "Desconexión" o el botón de "Atrás" de la barra de navegación inferior de Android. 2. Finaliza la conexión entre el móvil y el vehículo.
Postcondiciones	La aplicación se cierra y la conexión entre dispositivos finaliza.
Excepción	No aplica.

Tabla 38: Caso de uso CU_U_07

3.2.2. Casos de uso de vehículo

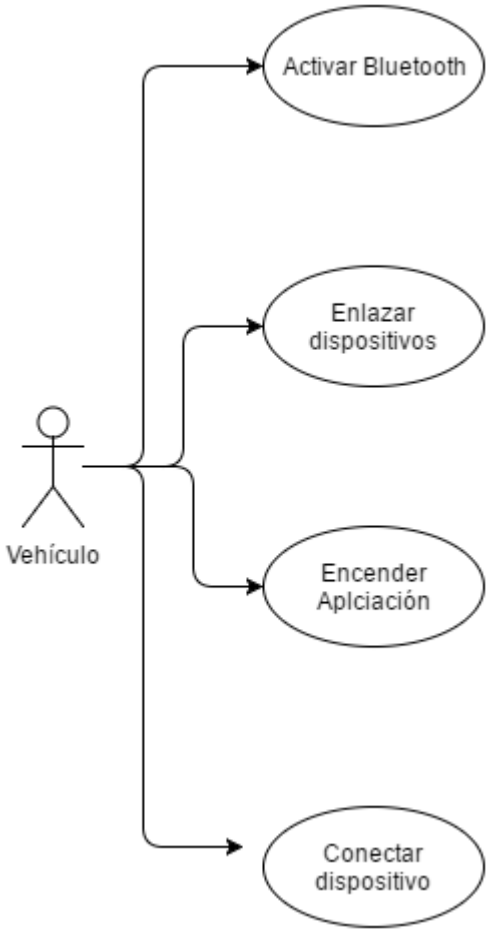


Ilustración 21: Casos de uso Vehículo

Identificador y Caso de uso	CU_V_01 – Recibir órdenes
Actores	Vehículo
Objetivo	El vehículo estará disponible para recibir las órdenes enviadas desde el teléfono por parte del usuario.
Precondiciones	Debe existir una conexión entre el móvil y el vehículo.
Escenario	1. El vehículo recibe una orden a través del módulo Bluetooth. 2. El módulo transfiere la orden al Arduino.
Postcondiciones	El Arduino recibe la orden para posteriormente poder procesarla y ejecutarla.
Excepción	1. El módulo Bluetooth no se encuentra disponible.

Tabla 39: Caso de uso CU_V_01

Identificador y Caso de uso	CU_V_02 – Procesar órdenes
Actores	Vehículo
Objetivo	El Arduino procesa las órdenes mediante el software cargado en él.
Precondiciones	1. Debe existir una conexión entre el móvil y el vehículo. 2. El Arduino debe tener cargado un software que funcione correctamente y tenga en cuenta las posibles órdenes que pueden ser enviadas.
Escenario	1. El vehículo recibe una orden a través del módulo Bluetooth. 2. El módulo transfiere la orden al Arduino. 3. El Arduino procesa la orden y selecciona qué acción debe realizar el vehículo.
Postcondiciones	El Arduino procesa la orden que llega para posteriormente poder ejecutarla.
Excepción	1. El módulo Bluetooth no se encuentra disponible. 3. La orden no existe, por lo que el vehículo no hace nada (error en el software).

Tabla 40: Caso de uso CU_V_02

Identificador y Caso de uso	CU_V_03 – Ejecutar órdenes
Actores	Vehículo
Objetivo	El Arduino ejecuta la orden que ha procesado para que el vehículo realice lo correspondiente.
Precondiciones	1. Debe existir una conexión entre el móvil y el vehículo. 2. La orden debe ser procesada correctamente, es decir, el software contempla dicha orden.
Escenario	1. El vehículo recibe una orden a través del módulo Bluetooth. 2. El módulo transfiere la orden al Arduino. 3. El Arduino procesa la orden y selecciona qué acción debe realizar el vehículo. 4. El vehículo realiza la acción pertinente.
Postcondiciones	Una vez ejecutada la orden, el vehículo puede: moverse hacia adelante, moverse hacia detrás, detenerse, girar a izquierda y derecha y moverse automáticamente.
Excepción	No aplica.

Tabla 41: Caso de uso CU_V_03

Identificador y Caso de uso	CU_V_04 – Sistema anti-colisiones
Actores	Vehículo.
Objetivo	El vehículo detectará un obstáculo y evitará colisionarse con él.
Precondiciones	Debe estar activado el modo de conducción automático.
Escenario	1. El vehículo avanza hacia adelante. 2. El vehículo detecta un obstáculo a una distancia determinada. 3. El vehículo se detiene un instante. 4. El vehículo gira a izquierda o a derecha de manera aleatoria durante un tiempo determinado. 5. El vehículo avanza hacia adelante hasta detectar otro obstáculo.
Postcondiciones	El vehículo tendrá activado el sistema anti-colisiones el tiempo que tenga activado el modo automático.
Excepción	No aplica.

Tabla 42: Caso de uso CU_V_04

3.3. Requisitos de sistema

3.3.1. Requisitos funcionales

RS-F-01	
Descripción:	El usuario podrá activar el Bluetooth, si se encuentra desactivado, desde la propia aplicación.
Necesidad:	Esencial
Claridad:	Alta
Prioridad:	Alta
Estabilidad:	Baja
Origen	RU-C-01
Verificabilidad:	Alta

Tabla 43: Requisito de sistema RS-F-01

RS-F-02	
Descripción:	No se podrá visualizar la lista de dispositivos vinculados sin tener activado el Bluetooth.
Necesidad:	Esencial
Claridad:	Alta
Prioridad:	Alta
Estabilidad:	Baja
Origen	RU-C-01
Verificabilidad:	Alta

Tabla 44:Requisito de sistema RS-F02

RS-F-03	
Descripción:	El dispositivo Bluetooth del vehículo y el dispositivo móvil deben estar vinculados antes de iniciar la aplicación.
Necesidad:	Esencial
Claridad:	Alta
Prioridad:	Alta
Estabilidad:	Baja
Origen	RU-C-02
Verificabilidad:	Alta

Tabla 45: Requisito de sistema RS-F-03

RS-F-04	
Descripción:	El dispositivo del vehículo, una vez vinculado correctamente con el teléfono, aparecerá en la lista de vinculados del teléfono.
Necesidad:	Esencial
Claridad:	Alta
Prioridad:	Alta
Estabilidad:	Baja
Origen	RU-C-02
Verificabilidad:	Alta

Tabla 46: Requisito de sistema RS-F-04

RS-F-05	
Descripción:	Se notificará al usuario que se ha conectado con el dispositivo del vehículo correctamente.
Necesidad:	Esencial
Claridad:	Alta
Prioridad:	Alta
Estabilidad:	Baja
Origen	RU-C-03
Verificabilidad:	Alta

Tabla 47: Requisito de sistema RS-F-05

RS-F-06	
Descripción:	En el caso de que no se pueda establecer una conexión, se notificará al usuario.
Necesidad:	Esencial
Claridad:	Alta
Prioridad:	Alta
Estabilidad:	Baja
Origen	RU-C-03
Verificabilidad:	Alta

Tabla 48: Requisito de sistema RS-F-06

RS-F-07	
Descripción:	El usuario podrá intentar conectarse de nuevo hasta que se consiga la conexión.
Necesidad:	Esencial
Claridad:	Alta
Prioridad:	Alta
Estabilidad:	Baja
Origen	RU-C-03
Verificabilidad:	Alta

Tabla 49: Requisito de sistema RS-F-07

RS-F-08	
Descripción:	El usuario deberá reintentar la conexión si el tiempo es superior a 15 segundos.
Necesidad:	Esencial
Claridad:	Alta
Prioridad:	Alta
Estabilidad:	Baja
Origen	RU-C-04
Verificabilidad:	Alta

Tabla 50: Requisito de sistema RS-F-08

RS-F-09	
Descripción:	Por defecto, al verificar que la conexión es correcta, se habilitarán los botones de control del vehículo.
Necesidad:	Esencial
Claridad:	Alta
Prioridad:	Alta
Estabilidad:	Baja
Origen	RU-C-05
Verificabilidad:	Alta

Tabla 51:Requisito de sistema RS-F-09

RS-F-10	
Descripción:	El botón de modo automático permanecerá habilitado mientras se encuentre activado el modo manual, mientras que el de modo manual permanecerá deshabilitado.
Necesidad:	Esencial
Claridad:	Alta
Prioridad:	Alta
Estabilidad:	Baja
Origen	RU-C-06
Verificabilidad:	Alta

Tabla 52: Requisito de sistema RS-F-10

RS-F-11	
Descripción:	El botón de modo manual permanecerá habilitado mientras se encuentre activado el modo automático, mientras que el de modo automático permanecerá deshabilitado.
Necesidad:	Esencial
Claridad:	Alta
Prioridad:	Alta
Estabilidad:	Baja
Origen	RU-C-06
Verificabilidad:	Alta

Tabla 53: Requisito de sistema RS-F-11

RS-F-12	
Descripción:	El usuario manejará el vehículo en modo manual mediante los botones que aparecerán en la aplicación (mover hacia adelante, hacia atrás, girar y detener).
Necesidad:	Esencial
Claridad:	Alta
Prioridad:	Alta
Estabilidad:	Baja
Origen	RU-C-07
Verificabilidad:	Alta

Tabla 54: Requisito de sistema RS-F-12

RS-F-13	
Descripción:	El usuario podrá detener el vehículo, mediante el botón de detener, únicamente si este se encuentra en movimiento. Si se encuentra detenido, el botón de detener se encontrará deshabilitado.
Necesidad:	Esencial
Claridad:	Alta
Prioridad:	Alta
Estabilidad:	Baja
Origen	RU-C-07
Verificabilidad:	Alta

Tabla 55: Requisito de sistema RS-F-13

RS-F-14	
Descripción:	El usuario podrá mover el vehículo, mediante los botones de avanzar o retroceder, únicamente si este se encuentra detenido. Si se encuentra en movimiento, los botones de avanzar o retroceder se encontrarán deshabilitados.
Necesidad:	Esencial
Claridad:	Alta
Prioridad:	Alta
Estabilidad:	Baja
Origen	RU-C-07
Verificabilidad:	Alta

Tabla 56:Requisito de sistema RS-F-14

RS-F-15	
Descripción:	El usuario podrá cambiar a modo automático pulsando el botón correspondiente.
Necesidad:	Esencial
Claridad:	Alta
Prioridad:	Alta
Estabilidad:	Baja
Origen	RU-C-07
Verificabilidad:	Alta

Tabla 57: Requisito de sistema RS-F-15

RS-F-16	
Descripción:	Los botones del manejo manual del vehículo permanecerán deshabilitados en el modo automático.
Necesidad:	Esencial
Claridad:	Alta
Prioridad:	Alta
Estabilidad:	Baja
Origen	RU-C-08
Verificabilidad:	Alta

Tabla 58: Requisito de sistema RS-F-16

RS-F-17	
Descripción:	El usuario podrá cambiar a modo manual pulsando el botón correspondiente.
Necesidad:	Esencial
Claridad:	Alta
Prioridad:	Alta
Estabilidad:	Baja
Origen	RU-C-08
Verificabilidad:	Alta

Tabla 59: Requisito de sistema RS-F-17

RS-F-18	
Descripción:	En la pantalla de manejo de la aplicación aparecerán todos los botones: conectar/desconectar dispositivo, modo manual y botones de cambio de modo.
Necesidad:	Esencial
Claridad:	Alta
Prioridad:	Alta
Estabilidad:	Baja
Origen	RU-C-09
Verificabilidad:	Alta

Tabla 60: Requisito de sistema RS-F-18

RS-F-19	
Descripción:	El usuario podrá finalizar la conexión Bluetooth con el vehículo desde la aplicación.
Necesidad:	Esencial
Claridad:	Alta
Prioridad:	Alta
Estabilidad:	Baja
Origen	RU-C-10
Verificabilidad:	Alta

Tabla 61:Requisito de sistema RS-F-19

RS-F-20	
Descripción:	El dispositivo Bluetooth del vehículo y el teléfono seguirán vinculados, aunque se finalice la conexión.
Necesidad:	Esencial
Claridad:	Alta
Prioridad:	Alta
Estabilidad:	Baja
Origen	RU-C-10
Verificabilidad:	Alta

Tabla 62: Requisito de sistema RS-F-20

RS-F-21	
Descripción:	El usuario será notificado ante cualquier error en la aplicación.
Necesidad:	Esencial
Claridad:	Alta
Prioridad:	Alta
Estabilidad:	Baja
Origen	RU-C-11
Verificabilidad:	Alta

Tabla 63: Requisito de sistema RS-F-21

RS-F-22	
Descripción:	El usuario deberá reiniciar la aplicación si el tiempo de respuesta es superior a 30 segundos.
Necesidad:	Esencial
Claridad:	Alta
Prioridad:	Alta
Estabilidad:	Baja
Origen	RU-C-12
Verificabilidad:	Alta

Tabla 64: Requisito de sistema RS-F-22

RS-F-23	
Descripción:	El vehículo poseerá un módulo Bluetooth para transmitir los datos entre la aplicación y el Arduino.
Necesidad:	Esencial
Claridad:	Alta
Prioridad:	Alta
Estabilidad:	Baja
Origen	RU-C-13
Verificabilidad:	Alta

Tabla 65: Requisito de sistema RS-F-23

RS-F-24	
Descripción:	El módulo Bluetooth del vehículo se podrá enlazar con uno o varios dispositivos móviles.
Necesidad:	Esencial
Claridad:	Alta
Prioridad:	Alta
Estabilidad:	Baja
Origen	RU-C-13
Verificabilidad:	Alta

Tabla 66: Requisito de sistema RS-F-24

RS-F-25	
Descripción:	El módulo Bluetooth del vehículo podrá estar conectado únicamente con un dispositivo móvil.
Necesidad:	Esencial
Claridad:	Alta
Prioridad:	Alta
Estabilidad:	Baja
Origen	RU-C-13
Verificabilidad:	Alta

Tabla 67: Requisito de sistema RS-F-25

RS-F-26	
Descripción:	El LED del módulo Bluetooth parpadeará si no existe ninguna conexión con un dispositivo.
Necesidad:	Esencial
Claridad:	Alta
Prioridad:	Alta
Estabilidad:	Baja
Origen	RU-C-13
Verificabilidad:	Alta

Tabla 68: Requisito de sistema RS-F-26

RS-F-27	
Descripción:	Una vez establecida la conexión con un dispositivo, el led del módulo Bluetooth dejará de parpadear y permanecerá fijo.
Necesidad:	Esencial
Claridad:	Alta
Prioridad:	Alta
Estabilidad:	Baja
Origen	RU-C-13
Verificabilidad:	Alta

Tabla 69: Requisito de sistema RS-F-27

RS-F-28	
Descripción:	Solamente será necesario emparejar los dispositivos una vez, no cada vez que se inicie la aplicación.
Necesidad:	Esencial
Claridad:	Alta
Prioridad:	Alta
Estabilidad:	Baja
Origen	RU-C-13
Verificabilidad:	Alta

Tabla 70: Requisito de sistema RS-F-28

RS-F-29	
Descripción:	El módulo Bluetooth deberá estar encendido para que aparezca como dispositivo disponible.
Necesidad:	Esencial
Claridad:	Alta
Prioridad:	Alta
Estabilidad:	Baja
Origen	RU-C-14
Verificabilidad:	Alta

Tabla 71: Requisito de sistema RS-F-29

RS-F-30	
Descripción:	El vehículo estará dotado de un sensor que calcule la distancia entre el vehículo y el obstáculo con el que se encuentre de frente.
Necesidad:	Esencial
Claridad:	Alta
Prioridad:	Alta
Estabilidad:	Baja
Origen	RU-C-15
Verificabilidad:	Alta

Tabla 72: Requisito de sistema RS-F-30

RS-F-31	
Descripción:	El sensor anti-colisiones se activará únicamente en el modo automático.
Necesidad:	Esencial
Claridad:	Alta
Prioridad:	Alta
Estabilidad:	Baja
Origen	RU-C-15
Verificabilidad:	Alta

Tabla 73: Requisito de sistema RS-F-31

RS-F-32	
Descripción:	Si el sensor detecta un obstáculo a una distancia determinada, el vehículo se detendrá automáticamente y no colisionará frontalmente.
Necesidad:	Esencial
Claridad:	Alta
Prioridad:	Alta
Estabilidad:	Baja
Origen	RU-C-15
Verificabilidad:	Alta

Tabla 74: Requisito de sistema RS-F-32

RS-F-33	
Descripción:	Se utilizará una pila de 9V como fuente de alimentación externa para el vehículo.
Necesidad:	Esencial
Claridad:	Alta
Prioridad:	Alta
Estabilidad:	Baja
Origen	RU-C-16
Verificabilidad:	Alta

Tabla 75: Requisito de sistema RS-F-33

RS-F-34	
Descripción:	El vehículo poseerá dos motores eléctricos de corriente continua para mover las ruedas.
Necesidad:	Esencial
Claridad:	Alta
Prioridad:	Alta
Estabilidad:	Baja
Origen	RU-C-17
Verificabilidad:	Alta

Tabla 76: Requisito de sistema RS-F-34

RS-F-35	
Descripción:	El vehículo poseerá además una “rueda loca” sin tracción para generar estabilidad en la estructura.
Necesidad:	Esencial
Claridad:	Alta
Prioridad:	Alta
Estabilidad:	Baja
Origen	RU-C-17
Verificabilidad:	Alta

Tabla 77: Requisito de sistema RS-F-35

RS-F-36	
Descripción:	El Arduino recibirá los datos enviados del módulo Bluetooth, que actuará como puerto serial.
Necesidad:	Esencial
Claridad:	Alta
Prioridad:	Alta
Estabilidad:	Baja
Origen	RU-C-18
Verificabilidad:	Alta

Tabla 78: Requisito de sistema RS-F-36

RS-F-37	
Descripción:	El software deberá implementar una lógica distinta dependiendo de los datos recibidos.
Necesidad:	Esencial
Claridad:	Alta
Prioridad:	Alta
Estabilidad:	Baja
Origen	RU-C-18
Verificabilidad:	Alta

Tabla 79: Requisito de sistema RS-F-37

RS-F-38	
Descripción:	El software debe compilar sin errores ni advertencias antes de cargarlo en el Arduino.
Necesidad:	Esencial
Claridad:	Alta
Prioridad:	Alta
Estabilidad:	Baja
Origen	RU-C-18
Verificabilidad:	Alta

Tabla 80: Requisito de sistema RS-F-38

RS-F-39	
Descripción:	Se utilizará una pila de 9V para alimentar el Arduino ya que debe funcionar de manera autónoma.
Necesidad:	Esencial
Claridad:	Alta
Prioridad:	Alta
Estabilidad:	Baja
Origen	RU-C-19
Verificabilidad:	Alta

Tabla 81: Requisito de sistema RS-F-39

3.3.2. Requisitos no funcionales

RS-NF-01	
Descripción:	La aplicación debe funcionar en la versión 6.0 de Android o inferior.
Necesidad:	Esencial
Claridad:	Alta
Prioridad:	Alta
Estabilidad:	Baja
Origen	RU-R-01
Verificabilidad:	Alta

Tabla 82:Requisito de sistema RS-NF-01

RS-NF-02	
Descripción:	La aplicación no está diseñada para tabletas o Smartphones con un tamaño superior a 5.5 pulgadas.
Necesidad:	Esencial
Claridad:	Alta
Prioridad:	Alta
Estabilidad:	Baja
Origen	RU-R-02
Verificabilidad:	Alta

Tabla 83: Requisito de sistema RS-NF-02

RS-NF-03	
Descripción:	La aplicación aparecerá completamente en castellano, con excepción de errores internos que podrían mostrarse en otro lenguaje.
Necesidad:	Esencial
Claridad:	Alta
Prioridad:	Alta
Estabilidad:	Baja
Origen	RU-R-03
Verificabilidad:	Alta

Tabla 84: Requisito de sistema RS-NF-03

RS-NF-04	
Descripción:	Se utilizará el lenguaje propio de Arduino, así como su API y sus librerías.
Necesidad:	Esencial
Claridad:	Alta
Prioridad:	Alta
Estabilidad:	Baja
Origen	RU-R-04
Verificabilidad:	Alta

Tabla 85: Requisito de sistema RS-NF-04

RS-NF-05	
Descripción:	Se utilizará un entorno de desarrollo a nivel de código, compatible para el sistema operativo Windows 10.
Necesidad:	Esencial
Claridad:	Alta
Prioridad:	Alta
Estabilidad:	Baja
Origen	RU-R-04
Verificabilidad:	Alta

Tabla 86: Requisito de sistema RS-NF-05

RS-NF-06	
Descripción:	Tanto el software como las conexiones hardware se realizarán utilizando únicamente un Arduino.
Necesidad:	Esencial
Claridad:	Alta
Prioridad:	Alta
Estabilidad:	Baja
Origen	RU-R-05
Verificabilidad:	Alta

Tabla 87: Requisito de sistema RS-NF-06

RS-NF-07	
Descripción:	El modelo de Arduino utilizado será “Arduino UNO”.
Necesidad:	Esencial
Claridad:	Alta
Prioridad:	Alta
Estabilidad:	Baja
Origen	RU-R-05
Verificabilidad:	Alta

Tabla 88:Requisito de sistema RS-NF-07

3.4. Matriz de trazabilidad entre requisitos

	RUC01	RUC02	RUC03	RUC04	RUC05	RUC06	RUC07	RUC08	RUC09	RUC10	RUC11	RUC12	RUC13	RUC14	RUC15	RUC16	RUC17	RUC18	RUC19	RUR01	RUR02	RUR03	RUR04	RUR05
RSF01	X																							
RSF02	X																							
RSF03		X																						
RSF04		X																						
RSF05			X																					
RSF06			X																					
RSF07			X																					
RSF08				X																				
RSF09					X																			
RSF10						X																		
RSF11						X																		
RSF12							X																	
RSF13							X																	
RSF14							X																	
RSF15							X																	
RSF16								X																
RSF17								X																
RSF18									X															
RSF19										X														
RSF20										X														
RSF21											X													
RSF22												X												
RSF23													X											
RSF24													X											
RSF25													X											
RSF26													X											
RSF27													X											
RSF28													X											
RSF29														X										
RSF30															X									
RSF31															X									
RSF32															X									
RSF33																X								
RSF34																	X							
RSF35																	X							
RSF36																		X						
RSF37																		X						
RSF38																		X						
RSF39																			X					
RSNF01																				X				
RSNF02																					X			
RSNF03																						X		
RSNF04																							X	
RSNF05																							X	
RSNF06																								X
RSNF07																								X

Tabla 89: Matriz de trazabilidad entre requisitos

Capítulo 4: Diseño del sistema

En este capítulo se realizará un diseño del sistema, tanto a nivel hardware como a nivel software. En el caso del hardware, se explicará el detalle el funcionamiento de cada componente que lo forma; para el software, se realizará un diseño tanto de la aplicación Android como de la aplicación de Arduino, con sus correspondientes diagramas de funcionamiento.

4.1. Diseño hardware

En este apartado se especificará el diseño por separado de todas las componentes hardware que forman el sistema, indicando qué función realizan y el funcionamiento detallado. Además, se incluirá un esquema eléctrico de las conexiones.

4.1.1. Receptor Bluetooth

El dispositivo utilizado para recibir los datos del teléfono móvil y transmitirlos al Arduino es el módulo Bluetooth HC-06.

Es importante recalcar que, para que el módulo HC-06 reciba correctamente los datos, las conexiones RX y TX deben ir cruzadas, y durante la carga del programa en el Arduino estas entradas deben desconectarse de la placa, ya que las entradas RX y TX son incompatibles con la entrada USB. Una vez cargado correctamente el programa, pueden conectarse.

Este módulo no necesita ninguna configuración previa, una vez conectado a tierra y al voltaje ya se encuentra disponible y visible para enlazar con cualquier dispositivo móvil. Si no se encuentra conectado a ningún dispositivo, el led que contiene aparecerá parpadeando. Una vez realizada la conexión entre el módulo y otro dispositivo, el led permanecerá fijo, hasta que se interrumpa de nuevo la conexión.

La configuración de los pines del módulo HC-06 con Arduino se muestran en la siguiente tabla:

HC-06	Arduino
GND	GND
VCC	5V
RXD	TX
TXD	RX

Tabla 90: Configuración pines HC-06

El esquema eléctrico es el siguiente:

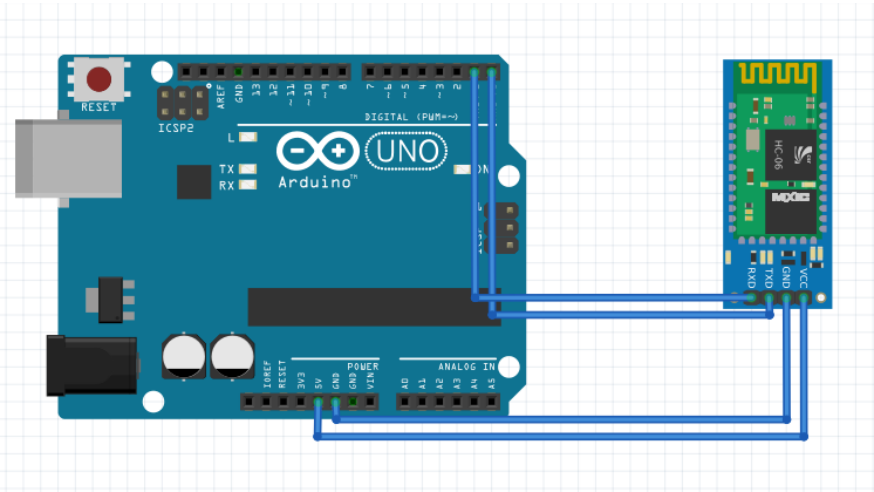


Ilustración 22: Esquema eléctrico HC-06

4.1.2. Sistema anti-colisiones

Como se ha explicado previamente, para diseñar el sistema anti-colisiones del vehículo se utilizará un sensor de ultrasonidos HC-SR04.

El sensor está formado por dos componentes: una de ellas envía un pulso de alta frecuencia, imperceptible por el ser humano, que rebota en un objeto que se encuentra próximo al sensor. Este pulso, al rebotar con el objeto, retorna al sensor de nuevo, y es recogido por la otra componente. De esta manera se puede determinar el tiempo de duración del pulso generado.

Hay que tener en cuenta que el pulso se envía a la velocidad del sonido, es decir, 343 m/s (en condiciones de 50% de humedad, 20ºC y presión atmosférica a nivel del mar). Realizando la transformación necesaria para determinar cuánto tiempo tarda en recorrer un centímetro, ya que será la unidad utilizada para medir la distancia, se obtiene el resultado aproximado de 29 microsegundos. La velocidad, por lo tanto, es 1/29 centímetros por microsegundo.

Recordando la definición de la velocidad, distancia/tiempo, se obtiene la distancia de la siguiente forma:

$$distancia = velocidad * tiempo$$

El tiempo hay que dividirlo entre 2, ya que el tiempo obtenido (en segundos) es el que ha tardado el pulso de ida y retorno.

Teniendo en cuenta los datos anteriores, la distancia (en cm) que recorre un pulso se calcula de la siguiente manera:

distancia = (tiempo / 2) / 29

El sensor HC-SR04 tiene dos pines: “Trig”, que es el que genera el pulso, y “Echo”, que es el que lo recibe.

Las configuraciones de los pines del sensor HC-SR04 y de Arduino se muestran en la siguiente tabla:

HC-SR04	Arduino
Gnd	Gnd
Echo	2
Trig	3
Vcc	5V

Tabla 91: Configuración de pines HC-SR04

El esquema eléctrico del sistema anti-colisiones es el siguiente:

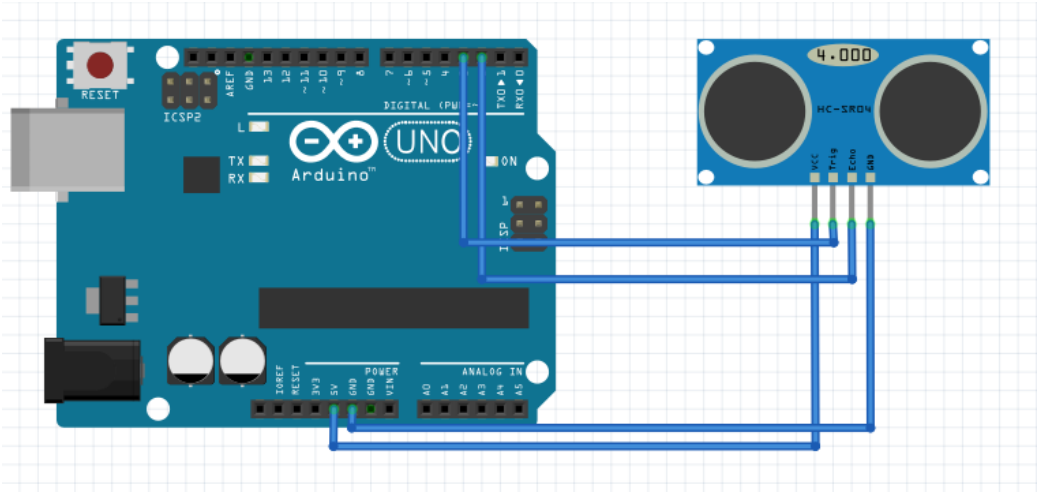


Ilustración 23: Esquema eléctrico HC-SR04

4.1.3. Puente H

Para producir movimiento en los motores del vehículo, es necesario realizar un puente H con la ayuda de un microchip L293D.

Un puente H es un circuito integrado que permite girar un motor eléctrico en ambos sentidos.

El chip L293D tiene 16 entradas; la siguiente imagen corresponde con el esquema del chip, y se indica la funcionalidad y las conexiones de los 16 pines.

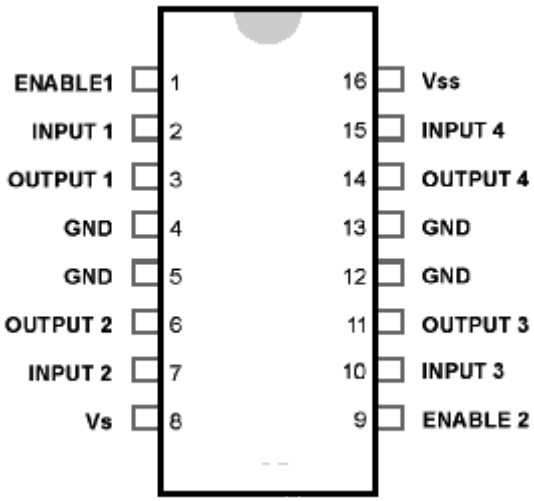


Ilustración 24: Esquema chip L293D

Los pines 1 y 9 se encargan de habilitar/deshabilitar los motores 1 y 2, respectivamente. Estos pines, al igual que el pin 16 (Vss), van conectados al voltaje proporcionado por el Arduino (5V).

Los pines 2 y 7 son los pines de entrada del primer motor, y están conectados a los pines digitales 5 y 6 del Arduino. Se encargan de controlar el giro del primer motor.

De manera análoga, los pines 10 y 15 son los pines de entrada del segundo motor, conectados a los pines digitales 9 y 10 del Arduino. Se encargan de controlar el giro del segundo motor.

Los pines 4, 5, 12 y 13 son conexiones a tierra (GND).

Los pines 3 y 6 van conectados al primer motor, y dependiendo de la lógica resultante con las entradas de los pines 2 y 7, el motor realizará un movimiento determinado, que será detallado más adelante. De la misma manera, los pines 11 y 14 van conectados al segundo motor.

Por último, el pin 8 (Vs) recibe la corriente proporcionada por la fuente de alimentación que va conectada a los motores. Este voltaje, a diferencia del pin 16 (Vss), puede variar desde 4,5 a 36V; en este caso, como ya se ha explicado, se utilizará una pila de 9V.

El funcionamiento del puente H en el circuito integrado L293D está basado en la lógica de un buffer triestado:

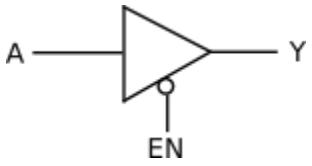


Ilustración 25: Buffer triestado

La entrada A puede estar en estado alto o bajo (1 o 0); EN es el pin de habilitación (Enable), y dependiendo del estado en que se encuentre, la salida Y puede tomar uno u otro valor. Si la entrada EN se encuentra en estado bajo (denotado como L o 0), la salida Y tomará un estado denominado alta impedancia, siendo la entrada A irrelevante.

Por el contrario, si EN se encuentra en estado alto (denotado como H o 1), es decir, el circuito está habilitado, la salida Y tomará el valor de A.

La siguiente tabla de funcionamiento hace referencia a las posibles situaciones mencionadas en las que puede encontrarse el buffer triestado:

TABLA DE FUNCIONAMIENTO
(para cada uno de los circuitos)

ENTRADAS †		SALIDA
A	EN	Y
H	H	H
L	H	L
X	L	Z

H = nivel alto L = nivel bajo X = irrelevante
Z = alta impedancia EN = habilitación
† en el modo de corte por protección térmica, las salidas estarán en el estado de alta impedancia, sin que afecte el estado de las entradas.

Tabla 92: Tabla funcionamiento Buffer triestado

La estructura lógica de cada motor en el integrado L293D, dado que cada motor tiene dos entradas, dos salidas y un pin de habilitación, y por tanto existen dos buffers triestado, es la siguiente:

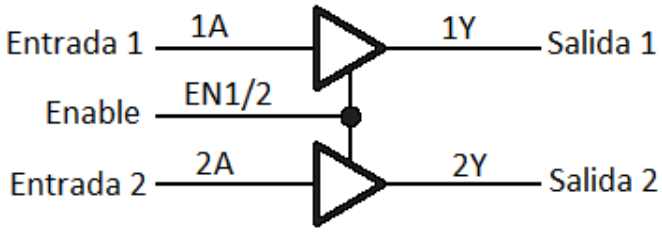


Ilustración 26: Esquema buffers triestado

El esquema del circuito del puente H para el motor 1 se detalla en la siguiente ilustración:

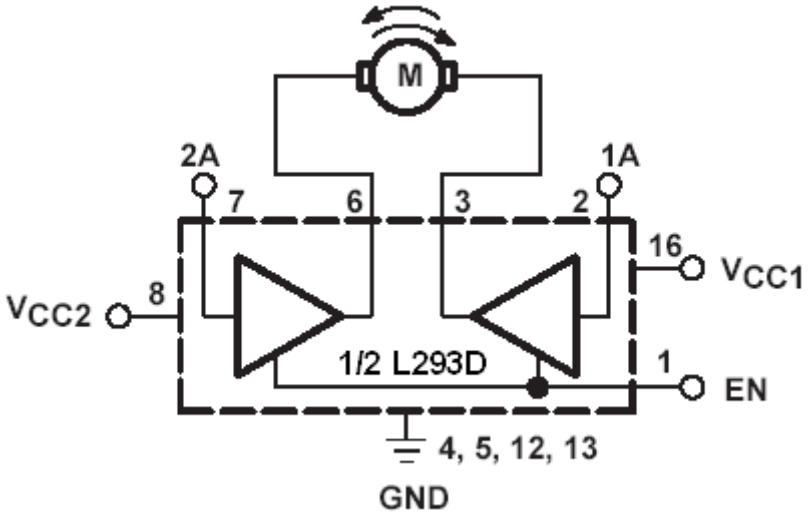


Ilustración 27: Puente H Motor

Las entradas 1A y 2A se producen por los pines 2 y 7, respectivamente. El pin de habilitación, EN, es el pin número 1. Las salidas correspondientes, tras aplicar la lógica anteriormente detallada del buffer triestado, se producen por los pines 3 y 6, conectados directamente al motor.

La siguiente tabla corresponde a la tabla de manejo del control del motor 1:

EN	1A	2A	FUNCIÓN
H	L	H	Giro a la derecha
H	H	L	Giro a la izquierda
H	L	L	Detención rápida
H	H	H	Detención rápida
L	X	X	Detención rápida

L = bajo, H = alto, X = no afecta

Tabla 93: Tabla de manejo Motor DC

Teniendo en cuenta los números de los pines de entrada, de salida y el pin de habilitación del segundo motor en el chip, el funcionamiento del motor 2 es el mismo que el que se ha detallado para el motor 1.

El esquema eléctrico de las conexiones de los dos motores y del chip L293D se muestra en la siguiente imagen:

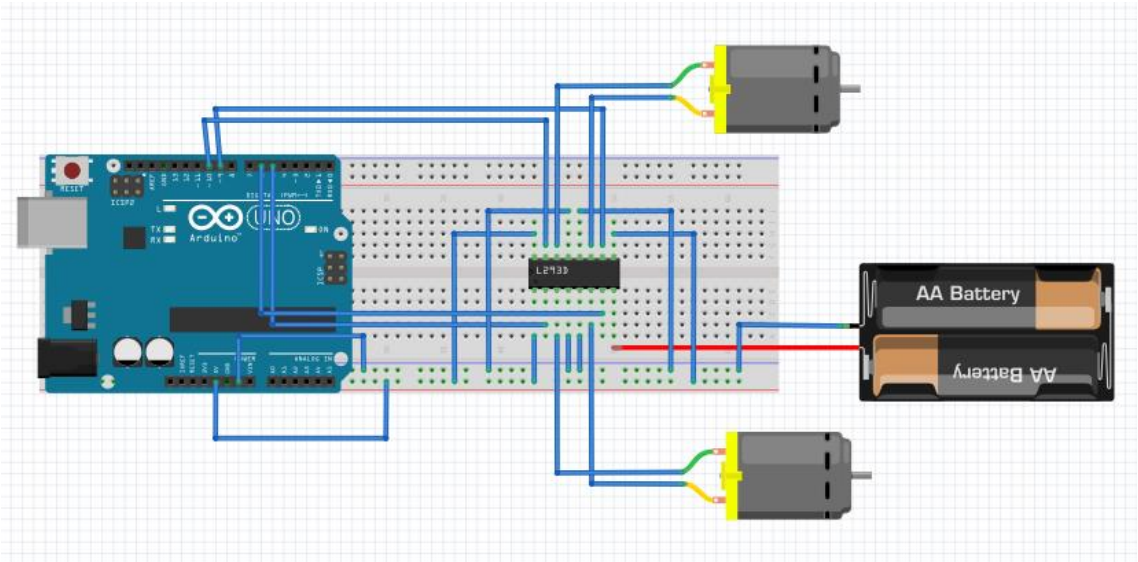


Ilustración 28: Esquema eléctrico puente H

4.1.4. Estructura del vehículo

Para realizar el montaje mecánico del prototipo de vehículo, es necesario seguir una serie de pasos.

Los materiales que se utilizarán son los siguientes:

- Estructura de metacrilato
- Dos ruedas con tracción
- Dos motores eléctricos de corriente continua y con reductor
- Una rueda loca
- Tornillos y arandelas de ajuste

El primer paso es ajustar los dos motores a la estructura de metacrilato a través de los tornillos y arandelas necesarios. Una vez queden fijos los motores, se añaden los reductores de motor y las ruedas. Este montaje debe quedar como muestra la siguiente imagen:

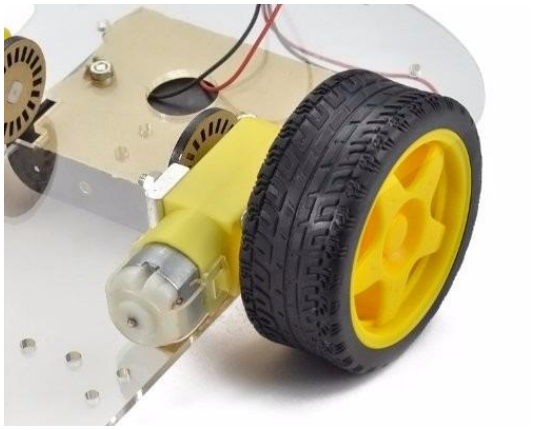


Ilustración 29: Montaje ruedas

El segundo paso será el montaje de la rueda loca, que debe ir atornillada en la parte posterior del vehículo. Esta rueda proporcionará además estabilidad a la estructura.

Ilustración 30: Montaje rueda loca

Finalmente, sobre la estructura se colocará la placa de pruebas, la placa de Arduino, las baterías que alimentan el Arduino y los motores y el sensor de ultrasonidos en la parte delantera, ya que de esta forma podrá detectar los obstáculos.

Una vez finalizado el diseño de cada componente que constituirá el vehículo, este ya estará disponible para agregarle la funcionalidad necesaria para que pueda realizar desplazamientos.

El diseño final de las componentes del vehículo se especifica en la siguiente imagen:

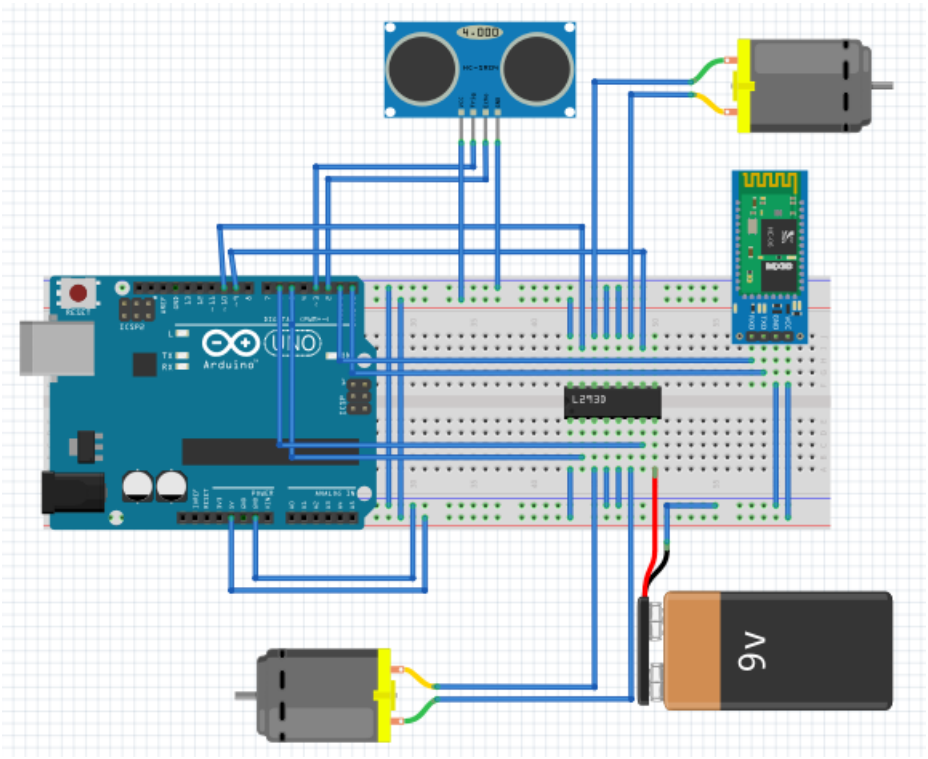


Ilustración 31: Esquema eléctrico vehículo

Para completar este diseño y poder implementar el sistema, es necesario incorporar el Arduino y la placa de pruebas con todas las conexiones al vehículo, además de agregar a los dos motores las ruedas y alimentar con otra pila el Arduino.

La funcionalidad necesaria para que el vehículo se pueda desplazar se explicará en el siguiente apartado del diseño y la implementación del software.

4.2. Diseño software

En este apartado se detallará el diseño del software del sistema, compuesto por la aplicación Android y el programa que irá cargado en la placa de Arduino. Una vez planteado el diseño, se procederá a explicar cómo ha sido programado cada software en el apartado de implementación.

4.2.1. Diseño de la aplicación Android

La aplicación estará formada por cuatro pantallas:

Primera pantalla: Inicio de la aplicación

Al iniciar la aplicación se muestra una pantalla de bienvenida que indica que la aplicación se está cargando, y finaliza pasado un tiempo determinado.

En ella aparece el nombre de la aplicación, My Arduino RC, junto al logotipo que ha sido elegido y un mensaje que indica al usuario que la aplicación se está iniciando.

Ilustración 32: Inicio aplicación

Segunda pantalla: Conexión de dispositivo

En esta pantalla aparece un mensaje de bienvenida junto a un icono, y ofrece al usuario la posibilidad de conectarse por Bluetooth a un dispositivo enlazado mediante un botón con el texto “Conectar Dispositivo” en la parte inferior.

Al pulsar sobre el botón pueden ocurrir dos casos:

- El usuario no tiene encendido el Bluetooth del teléfono; entonces aparecerá un mensaje de petición para que el usuario active el Bluetooth desde la aplicación, donde puede aceptar o denegar. En ambos casos, se redirige al usuario a la siguiente pantalla.

- El usuario tiene encendido el Bluetooth, por lo que el mensaje anterior no aparecerá, y se redirigirá directamente a la siguiente pantalla.

Ilustración 33: Conexión dispositivo

Ilustración 34: Permiso activación Bluetooth

Tercera pantalla: Listado de dispositivos enlazados

Esta pantalla aparecerá bien tenga el usuario activado el Bluetooth en el teléfono o no; si el usuario tiene activado el Bluetooth, al pulsar en el botón inferior de "Lista de dispositivos enlazados" podrá visualizar una lista en la que aparecen los dispositivos que el usuario tiene vinculados con su teléfono.

En caso contrario, si el usuario no tiene activado el Bluetooth, al pulsar en el botón no aparecerá ningún dispositivo, sino que aparecerá un mensaje de error donde avisa al usuario de que no existe ningún dispositivo disponible.

En este último caso, el usuario puede retroceder a la pantalla anterior, donde el usuario puede aceptar el permiso de activación de Bluetooth y entonces ya visualizar la lista, o bien en ese momento activar el Bluetooth desde la barra de notificaciones de Android, y volver a pulsar en el botón del listado.

Si el dispositivo no se encuentra en la lista, es posible que no se haya vinculado correctamente al teléfono; en ese caso, el usuario debe salir de la aplicación, ir a la configuración del Bluetooth del teléfono y emparejarlo con otro dispositivo.

Una vez se seleccione el dispositivo correspondiente al del vehículo, y este se encuentre disponible, se pasará automáticamente a la siguiente pantalla.

Si no se puede conectar con el dispositivo, automáticamente se saldrá de la aplicación, y el usuario deberá revisar que el dispositivo se encuentre disponible la siguiente vez que inicie la aplicación.

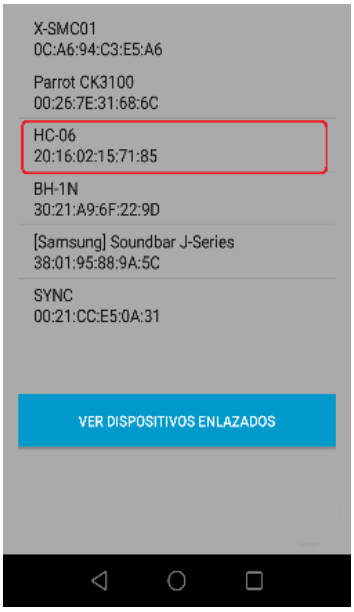


Ilustración 35: Dispositivos enlazados

Cuarta pantalla: Control del vehículo y desconexión

Una vez seleccionado el dispositivo enlazado y se realiza correctamente la conexión Bluetooth, se redirige a la última pantalla de la aplicación. En esta pantalla el usuario puede manejar el vehículo y cambiar el modo gracias a los botones correspondientes.

En la parte superior de la pantalla aparecen los botones de modo, donde el usuario inicialmente puede seleccionar el modo automático ya que, por defecto, al iniciar la aplicación, el control es manual.

Si el usuario selecciona el modo automático, los botones del control del vehículo permanecerán deshabilitados, y al pulsar en el botón de modo manual, se vuelven a habilitar.

En la parte inferior de la pantalla aparece el botón de desconectar, que al pulsarlo detiene la conexión entre el vehículo y el móvil y sale de la aplicación.

Ilustración 36: Control del vehículo

A continuación, se muestra el diagrama de flujo correspondiente al diseño de la aplicación:

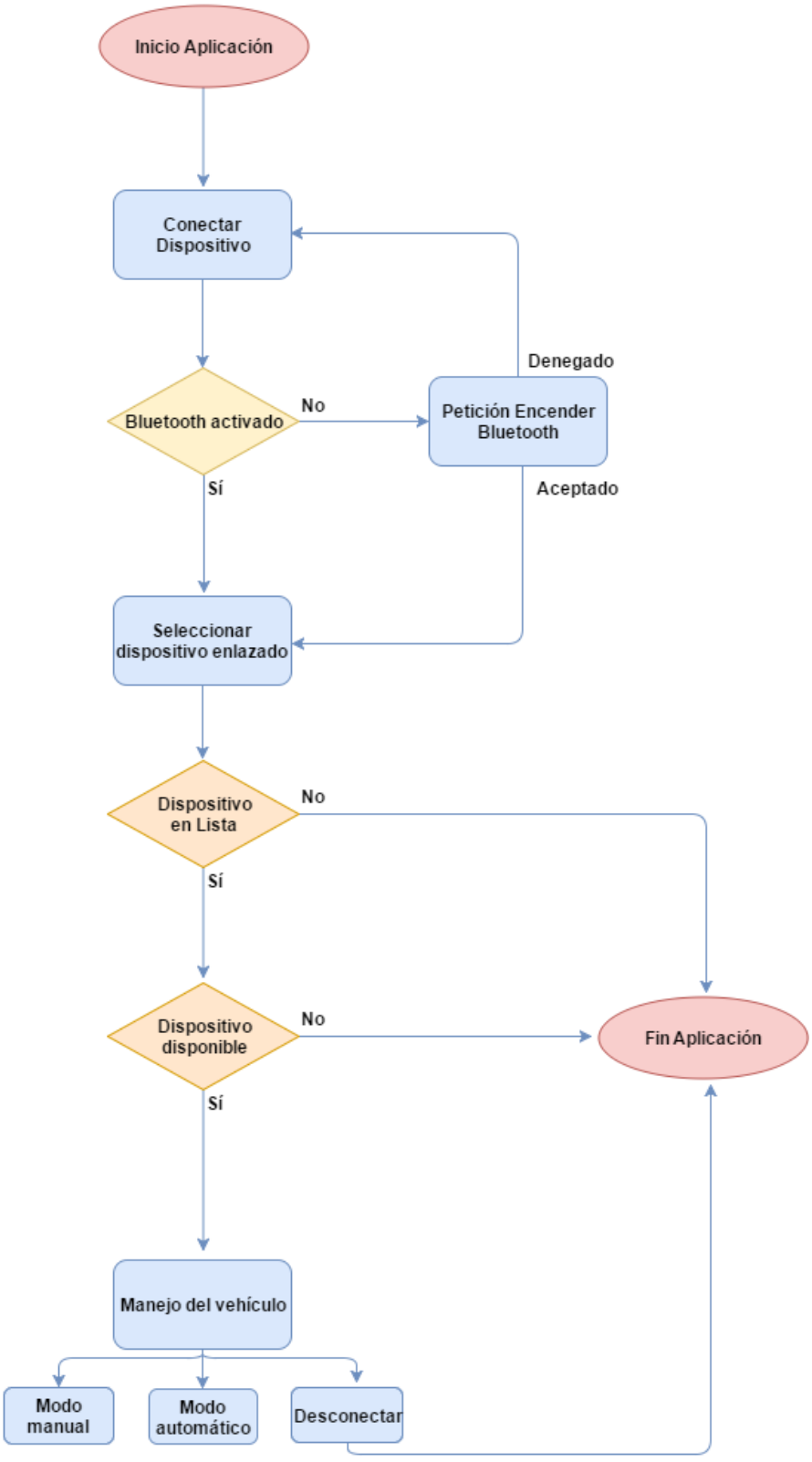


Ilustración 37: Diagrama de flujo aplicación Android

Tras realizar el diagrama de flujo correspondiente al diseño de la aplicación, se muestra la secuencia de pantallas por las que transcurre la aplicación, en el orden en el que se han detallado anteriormente:

Ilustración 38: Secuencia de pantallas de aplicación

4.2.2. Diseño del software de Arduino

El software de Arduino está estructurado en dos partes principales: la declaración y configuración de los pines y la ejecución de las instrucciones.

Primero se asignan valores enteros a las diferentes variables correspondientes con los pines de Arduino, y se crean otras variables que serán usadas en el transcurso de la aplicación.

La función setup() se encarga de inicializar el puerto serial, para que sea posible la comunicación entre el teléfono y el Arduino, y de configurar los pines declarados anteriormente como pines de entrada o de salida. La función setup() se ejecuta una única vez al iniciar el programa, antes de pasar a la segunda función loop().

La función loop() realiza la ejecución de las instrucciones del programa un número ilimitado de veces, ya que se trata de un bucle infinito, del que únicamente se podría salir bajo una cierta condición. En el caso de la aplicación para este sistema, no existe ninguna condición de salida.

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

La configuración de los pines de Arduino necesarios para el programa se muestran en la siguiente tabla:

Funcionalidad	Pin Arduino	Entrada / Salida
Entrada 1 motor 1	5	SALIDA
Entrada 2 motor 1	6	SALIDA
Entrada 1 motor 2	9	SALIDA
Entrada 2 motor 2	10	SALIDA
Módulo ultrasónico	3	SALIDA
Módulo ultrasónico	2	ENTRADA

Tabla 94: Configuración pines Arduino

La comunicación por el puerto serial se realiza a través del módulo Bluetooth, que es el encargado de recibir los datos procedentes del teléfono para enviárselos al Arduino. No es necesario declarar los pines ni realizar ninguna configuración, ya que los pines utilizados ya están configurados para enviar y recibir datos.

Las conexiones de pines entre Arduino y el módulo Bluetooth, sin tener en cuenta voltaje y tierra, se muestran en la siguiente tabla:

Pin Arduino	Pin HC-06
0 (RX)	TX
1 (TX)	RX

Tabla 95: Equivalencia de pines Arduino-HC06

Una vez configurados los pines, se pasa a la ejecución en modo bucle infinito del programa, donde el Arduino comienza a recibir datos a través del puerto serial. Estos datos son caracteres que se envían desde la aplicación móvil al pulsar los botones correspondientes.

Dependiendo del carácter recibido en cada iteración del bucle, el vehículo realizará una operación determinada durante un periodo de tiempo, y durará hasta que el usuario envíe otra orden distinta.

En la siguiente tabla se indica la correspondencia entre los posibles caracteres que el Arduino puede recibir y qué acción realiza el vehículo en cada caso:

Carácter recibido	Acción realizada
'a'	Movimiento hacia adelante
'b'	Movimiento hacia atrás
'r'	Giro a la derecha
'l'	Giro a la izquierda
's'	Detención
'X'	Cambio a modo manual
'Y'	Cambio a modo automático
Otro	No aplica

Tabla 96: Tabla de manejo de caracteres

Dado que en la pantalla principal de la aplicación hay únicamente siete botones, en el Arduino solamente se podrán procesar siete órdenes diferentes, por lo que, en teoría, no se puede recibir otro carácter que no sea alguno de los detallados en la tabla. En caso de que exista algún error en el puerto serial y se reciba otro carácter, no se contemplaría ninguna acción.

Movimiento hacia adelante, hacia atrás y detención

Los movimientos hacia adelante y atrás están basados en el control del giro de los motores, es decir, en el puente H. Recordando el puente H detallado en el apartado de diseño hardware de este capítulo, tenemos que los posibles giros de un motor son los siguientes:

EN	1A	2A	FUNCIÓN
H	L	H	Giro a la derecha
H	H	L	Giro a la izquierda
H	L	L	Detención rápida

Tabla 97: Tabla manejo puente H

Extrapolando estos conceptos a los dos motores, tenemos lo siguiente:

- Si los dos motores giran a la derecha, el vehículo se moverá hacia adelante.
- Si los dos motores giran a la izquierda, el vehículo se moverá hacia atrás.
- Si los dos motores no giran, el vehículo se detendrá.

La velocidad de movimiento, tanto hacia adelante como atarás, deberá ser constante.

Giros a izquierda y a derecha

El cambio de dirección se realizará únicamente cuando el movimiento del vehículo sea hacia adelante. La lógica de los giros también está basada en el puente H:

- Para girar a la derecha, el motor que proporciona tracción a la rueda izquierda se detiene durante un tiempo, mientras que el motor derecho se moverá hacia adelante.
- De forma análoga, para girar a la izquierda el motor derecho se detiene durante un tiempo y el motor izquierdo se mueve hacia adelante.

El tiempo que permanece un motor parado deberá ser un tiempo corto, de aproximadamente 300 milisegundos, para que el cambio de dirección no sea demasiado brusco. En el caso que el usuario desee un cambio de dirección más pronunciado, deberá pulsar varias veces el botón de la dirección deseada.

Una vez pasados los 300 milisegundos, el vehículo continuará la marcha hacia adelante.

Modo automático

Cuando el usuario selecciona el modo automático, el vehículo se desplazará hacia adelante, y se activará la funcionalidad del sistema anti-colisiones, es decir, el módulo ultrasónico empezará a calcular distancias frontales en intervalos de tiempo muy cortos, para mayor precisión.

En el momento que el sensor detecte un obstáculo a una distancia de 20 centímetros, el vehículo se detendrá, y girará automáticamente hacia uno de los dos lados. La decisión de giro a izquierda o a derecha será aleatoria, gracias a una operación matemática que entrará en funcionamiento en el momento de la detección del obstáculo.

Una vez gire el vehículo, continuará su marcha hacia adelante, hasta el momento en el que detecte otro obstáculo; entonces se repetirá el algoritmo de nuevo.

Si el vehículo no detecta ningún objeto a 20 centímetros, es decir, la distancia que recibe Arduino es superior a 20, no se detendrá y continuará su movimiento hacia adelante.

Modo manual

Si el usuario decide cambiar a modo manual, el vehículo se detendrá inmediatamente, y de nuevo podrá manejarlo a su antojo o cambiar de modo. El vehículo permanecerá detenido hasta que Arduino reciba un carácter por el puerto serial.

El siguiente diagrama de flujo muestra el funcionamiento del programa:

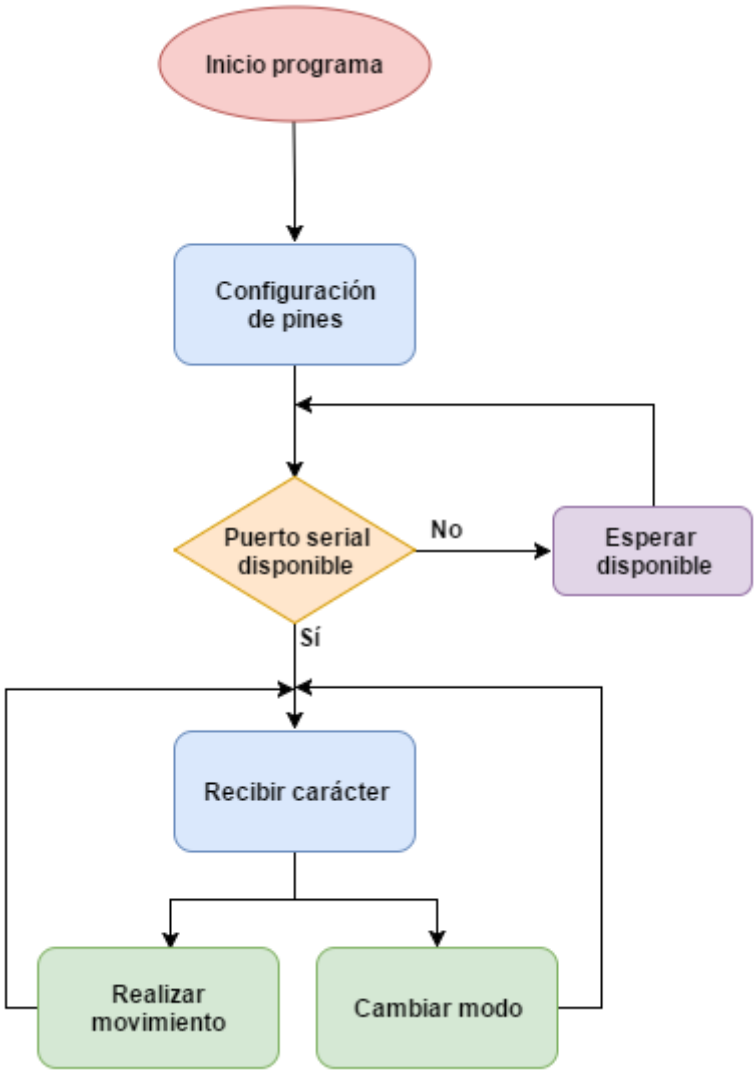


Ilustración 39: Diagrama de flujo software Arduino

4.3. Proceso de comunicación del sistema

Como conclusión, se resume el proceso mediante el cual el dispositivo móvil se conecta con Arduino y este es capaz de ejecutar las órdenes que le envía.

La siguiente imagen muestra el proceso que es llevado a cabo cuando el usuario utiliza la aplicación:

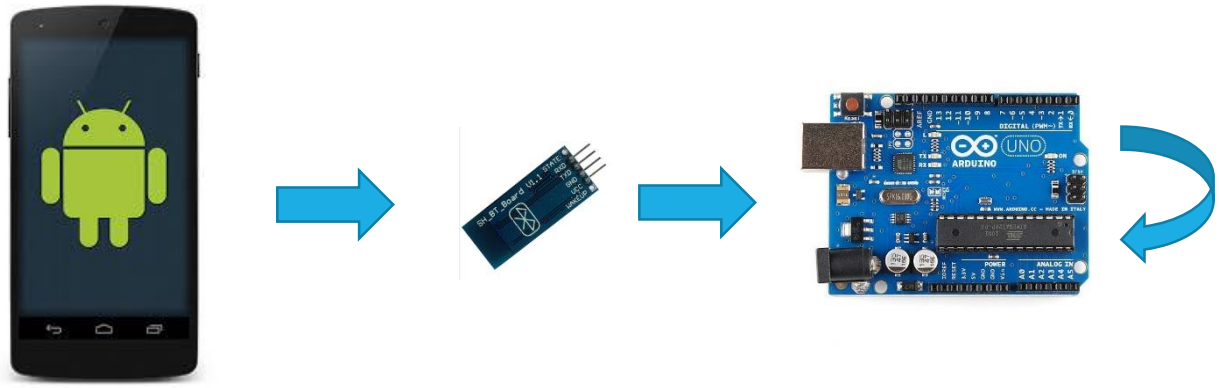


Ilustración 40: Comunicación Software-Hardware

El usuario, una vez iniciada la aplicación correctamente y situado en la pantalla de manejo del vehículo, envía órdenes para que el vehículo se mueva. Estas órdenes las recoge el receptor o módulo Bluetooth en forma de caracteres (ver diseño del software).

El módulo Bluetooth, que actúa de puerto serial, envía dicho carácter al Arduino, ubicado en el vehículo. Arduino recoge el carácter y lo procesa; dependiendo del carácter, el vehículo realiza una acción u otra. El vehículo continuará realizando dicha acción hasta que el usuario envíe otra orden.

En el momento que el usuario desee salir de la aplicación, el Arduino dejará de recibir datos ya que la conexión entre el teléfono y Arduino finalizará.

Capítulo 5: Implementación y pruebas del sistema

El propósito de este capítulo es realizar una implementación del diseño descrito en el capítulo anterior y crear un entorno de pruebas, con el objetivo de analizar el funcionamiento del sistema y verificar si todo se encuentra correcto o no.

5.1. Implementación del sistema

5.1.1. Implementación de la aplicación Android

La aplicación estará formada por cuatro actividades o clases Java, una por cada pantalla que aparece en la aplicación. A continuación, se detallará, en el orden descrito en el diseño, la funcionalidad descrita de cada pantalla.

FirstActivity.java

Esta clase corresponde a la primera pantalla de la aplicación; esta pantalla tendrá una duración de cuatro segundos, y solamente aparecerá cuando se inicie la aplicación.

Primero se define la variable TIEMPO y se le asigna un valor de 4000, que son los milisegundos que durará la aplicación.

```
public static int TIEMPO = 4000;
```

En el método onCreate de la clase, se crea un manejador (Handler) que será el encargado de establecer la duración de la actividad el tiempo deseado. En el método run(), propio del manejador, se creará un Intent para que cambie de la actividad actual a la siguiente, que será ConnectionMenu.

```
Handler handler = new Handler();
handler.postDelayed(new Runnable() {
    @Override
    public void run() {
        Intent intent = new Intent(FirstActivity.this,ConnectionMenu.class);
        startActivity(intent);
        finish();
    }
},TIEMPO);
```

ConnectionMenu.java

La funcionalidad de esta actividad consiste únicamente en crear un botón que, al pulsar sobre él, redirija al usuario a la siguiente actividad.

Para ello, en la clase Java de la actividad se crea una variable de tipo botón (Button) y se le asigna el valor del identificador de dicho botón, creado en el fichero xml del diseño, mediante la función findViewById:

```
Button connect;

connect = (Button) findViewById(R.id.connectButton);
```

Para implementar la funcionalidad que se realizará al pulsar en el botón, se crea un OnClickListener, que contiene el método onClick(), en el que se crea un Intent que cambie de una actividad a otra. En este caso, pasará a la actividad de seleccionar el dispositivo con el que conectarse.

```
connect.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intent = new Intent(ConnectionMenu.this,ListPaired.class);
        startActivity(intent);
    }
});
```

El botón “Conectar dispositivo” está implementado en formato xml de la siguiente manera:

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Conectar dispositivo"
    android:id="@+id/connectButton"
    android:layout_below="@+id/imageView2"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="60dp"
    android:textColor="@android:color/white"
    android:background="@android:color/holo_blue_dark"/>
```

ListPaired.java

En esta actividad el usuario seleccionará un dispositivo enlazado con el teléfono con el que conectarse.

Antes de mostrar los dispositivos vinculados al usuario, se realizan dos comprobaciones: que el dispositivo admita conectividad Bluetooth y, si la admite, que el usuario tenga encendido el Bluetooth.

Para ello se crea una variable de tipo BluetoothAdapter, se asigna el valor por defecto, que es getDefaultAdapter(), y se verifica que sea distinto del valor "null". Si el valor es null, la aplicación se finaliza. Si no es null se verifica que el Bluetooth esté activado, y en caso de que no se encuentre activado, se crea un Intent de petición al usuario para que active Bluetooth.

```
private BluetoothAdapter myBluetooth = null;
```

```
myBluetooth = BluetoothAdapter.getDefaultAdapter();

if (myBluetooth == null) {
    //Si no admite Bluetooth, finaliza aplicacion
    Toast.makeText(getApplicationContext(), "Bluetooth no disponible",
Toast.LENGTH_LONG).show();
    finish();
} else if (!myBluetooth.isEnabled()) {
    //Peticion de encender Bletooth
    Intent turnBTon = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
    startActivityForResult(turnBTon, 1);
}
```

Una vez encendido el Bluetooth del teléfono, se procede a listar los dispositivos vinculados para que el usuario seleccione uno de ellos.

```
paired.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        vinculados();
    }
});
```

Se crea la función vinculados(), que es la encargada de obtener la lista de dispositivos vinculados al teléfono.

```
private Set<BluetoothDevice> dispVinculados;


public void vinculados(){

   dispVinculados = myBluetooth.getBondedDevices();
    if (dispVinculados.size() > 0) {
        list = new ArrayList<>();
        for (BluetoothDevice device : dispVinculados) {
            //Obtenemos los nombres y direcciones MAC de los dispositivos
vinculados
            list.add(device.getName() + "\n" + device.getAddress());
        }
    } else {
        Toast.makeText(getApplicationContext(), "No existen dispositivos
vinculados", Toast.LENGTH_SHORT).show();
    }
    adapter = new ArrayAdapter<BluetoothDevice>(this,
android.R.layout.simple_list_item_1, list);
    listPaired.setAdapter(adapter);
    listPaired.setOnItemClickListener(myListClickListener);
}
```

Primero se crea una variable de tipo Set, que hace referencia a un objeto, en este caso de tipo BluetoothDevice (dispositivo Bluetooth). La función getBondedDevices obtiene todos los dispositivos vinculados a un objeto de tipo BluetoothAdapter.

Si existen dispositivos vinculados, es decir, el tamaño de la lista es mayor que 0, obtiene los nombres y direcciones MAC de los dispositivos, y los añade a un ArrayList, para posteriormente crear un ArrayAdapter, que será la lista de los dispositivos que se mostrará al usuario por pantalla.

El usuario seleccionará un dispositivo pulsando sobre él, momento en el que se ejecutará la función onItemClick, propia de un ArrayAdapter. En ella se recoje la dirección MAC del dispositivo seleccionado para pasarla, mediante un Intent, a la siguiente actividad, que servirá para realizar la conexión.

```
public void onItemClick(AdapterView<?> av, View v, int arg2, long arg3) {

    // Direccion MAC
    String info = ((TextView) v).getText().toString();
    String address = info.substring(info.length() - 17);

    // Intent para pasar a otra actividad
    Intent intent = new Intent(ListPaired.this, CarController.class);
    intent.putExtra(EXTRA_DEVICE_ADDRESS, address);
    startActivity(intent);
}
```

CarController.java

Primero se definen los botones que contendrá la pantalla:

```
go = (ImageButton) findViewById(R.id.goButton);
back = (ImageButton) findViewById(R.id.backButton);
left = (ImageButton) findViewById(R.id.leftButton);
right = (ImageButton) findViewById(R.id.rightButton);
stop = (ImageButton) findViewById(R.id.stopButton);
auto = (Button) findViewById(R.id.autoButton);
manual = (Button) findViewById(R.id.manualButton);
disconnect = (Button) findViewById(R.id.discButton);
```

En el método onResume() se recoge la dirección MAC del dispositivo con el que se va a realizar la conexión, y se crea un objeto de tipo BluetoothDevice con dicha dirección:

```
Intent intent = getIntent();
address = intent.getStringExtra(ListPaired.EXTRA_DEVICE_ADDRESS);

BluetoothDevice device = btAdapter.getRemoteDevice(address);
```

Para realizar las comunicaciones por Bluetooth hay que crear un socket y conectarse, manejando las posibles excepciones que puedan ocurrir:

```
BluetoothSocket btSocket = null;
try{
    btSocket = createSocket(device);
} catch (IOException e) {
    Toast.makeText(getBaseContext(), "Error al crear el socket",
Toast.LENGTH_LONG).show();
}

try{
    btSocket.connect();
} catch (IOException e){
    try{
        btSocket.close();
    } catch (IOException e1){
        Toast.makeText(getBaseContext(), "Error al cerrar el socket",
Toast.LENGTH_LONG).show();
    }
}
```

El método createSocket(device) crea un socket con un dispositivo Bluetooth con un UUID, que identifica el servicio Bluetooth que se usa en la conexión. Este concepto es similar a los números de puerto en Internet.

Por defecto se usa el siguiente UUID, que identifica a la mayoría de servicios Bluetooth, y que es válido para comunicaciones seriales con Arduino:

```
private static final UUID myUUID = UUID.fromString("00001101-0000-1000-8000-
00805F9B34FB");
```

El método que crea el socket sería, por tanto:

```
private BluetoothSocket createSocket(BluetoothDevice device) throws
IOException{
    return device.createRfcommSocketToServiceRecord(myUUID);
}
```

Una vez establecida la comunicación por el socket, se implementa la funcionalidad de los botones, creando para cada botón un OnClickListener que contiene el método OnClick(). En este método se implementa el siguiente código:

```
if(btSocket!=null){
    try{
        mConnectedThread.write("N");
    }catch (Exception e){

Toast.makeText(getApplicationContext(),"Error",Toast.LENGTH_SHORT).show();
    }
}
```

Donde `mConnectedThread` es un objeto de tipo Thread mediante el cual se envía el carácter “N” con el método write, para que sea transmitido de manera serial al Arduino.

5.1.2. Implementación del software de Arduino

La implementación del programa cargado en la placa Arduino consta de tres partes: definición de variables, configuración e inicialización de componentes e implementación de las funcionalidades.

Definición de variables

Las variables utilizadas en el programa se definen al principio. El valor de una variable puede ser un valor específico o un pin de Arduino.

En el diseño de la aplicación, observando la tabla “”, correspondiente a la configuración de pines de Arduino, las variables que adquieren valores de pines son las correspondientes a los motores y las del sensor ultrasónico.

```
int motor11 = 5;
int motor12 = 6;
int motor21 = 9;
int motor22 = 10;
int echo = 2;
int trig = 3;
```

El resto de variables corresponden al estado, que es el carácter que el Arduino recibe, velocidad del vehículo, distancia, duración y una variable entera que actuará de número aleatorio más adelante.

```
int vel = 40;
char estado = 'g';
int duracion, distancia;
int r;
```

Configuración de componentes

La configuración de las componentes se realiza en la función void setup(). Se configura como de entrada o de salida las variables anteriores que hagan referencia a pines de Arduino, y también se inicializa el puerto serial con una velocidad por defecto de 9600 baudios:

```
void setup() {
  Serial.begin(9600);
  pinMode(motor11, OUTPUT);
  pinMode(motor12, OUTPUT);
  pinMode(motor21, OUTPUT);
  pinMode(motor22, OUTPUT);
  pinMode(echo, INPUT);
  pinMode(trig, OUTPUT);
}
```

Como se puede observar todos los pines son de salida excepto el pin “echo”, que es el pin de módulo ultrasónico encargado de medir el tiempo que tarda el ultrasonido, por lo tanto, debe ser un pin de entrada.

Nota: la notación seguida para los motores es motorXY, donde X es el número del motor (1 o 2) e Y indica el pin de entrada del motor.

Implementación de las funcionalidades

El primer paso que realiza la función es leer del puerto serial el carácter obtenido y almacenarlo en la variable de tipo carácter “estado”.

```
if(Serial.available()>0){
  estado = Serial.read();
}
```

Posteriormente, dependiendo del valor del carácter recibido (ver tabla “”), el vehículo realizará o un determinado movimiento o un cambio de modo.

La función analogWrite(a,b) permite escribir en el pin marcado como “a” el valor “b”. En este caso el pin será el correspondiente al del motor y el valor será la velocidad declarada al principio.

Para que el vehículo se desplace hacia adelante, el Arduino debe recibir por el puerto serial el carácter ‘a’. Recordando el diseño del puente H, los motores se mueven hacia adelante si uno de los dos pines de entrada de cada motor se encuentra en estado HIGH o 1 (suponiendo que el pin de habilitación de cada motor está activado y conectado a corriente).

```
if(estado=='a'){
    analogWrite(motor11, vel);
    analogWrite(motor21, vel);
    analogWrite(motor12, 0);
    analogWrite(motor22, 0);
}
```

La implementación del modo automático comienza cuando el usuario pulsa el botón desde la aplicación y el Arduino recibe el carácter 'Y'.

En ese momento, el vehículo comienza a desplazarse al frente y entra en funcionamiento el sensor ultrasónico, que envía cada 200 milisegundos un ultrasonido midiendo el tiempo que tarda.

Primero se calibra el ultrasónico implementando el siguiente código:

```
digitalWrite(trig, HIGH);
delay(0.01);
digitalWrite(trig, LOW);
```

Utilizando la fórmula descrita en el diseño para calcular la distancia a partir del tiempo, calcula la distancia en centímetros y la almacena en la variable "distancia". Si la distancia es inferior a 20 centímetros, una distancia prudencial para que el vehículo se detenga sin colisionar, el vehículo se detiene. Es decir, se escribe un 0 en todos los pines de los motores.

```
duracion = pulseIn(echo, HIGH);
distancia = (duracion/2) / 29;
Serial.println(distancia);
delay(200);

if (distancia <= 20 && distancia >=2){

   analogWrite(motor11, 0);
   analogWrite(motor21, 0);
   analogWrite(motor12, 0);
   analogWrite(motor22, 0);
```

Para determinar el giro posterior a la detención por el encuentro de un obstáculo, se implementa un algoritmo basado en una función aleatoria; se asigna a la variable "r" un número aleatorio comprendido entre 1 y 10, mediante la función random(x,y).

Esta función, propia del API de Arduino, genera un número aleatorio comprendido entre los valores x e y.

```
r = random(1,10);
```

El algoritmo determina si el valor de "r" es par o impar, esto significa que si el resto de dividir el número entre 2 da como resultado 0, el número es par, y en caso contrario el número es impar.

Al igual que en otros lenguajes de programación, como Java, el resto de dividir un número entre otro se realiza mediante el operando "%".

Si el número es par, el vehículo girará a la izquierda, y si es impar girará a la derecha. Una vez realizado el giro, se detendrá un tiempo de 500 milisegundos y continuará el movimiento hacia adelante.

```
if(r%2 == 0){
  analogWrite(motor11, 0);
  analogWrite(motor21, vel);
  analogWrite(motor12, 0);
  analogWrite(motor22, 0);
  delay(500);
} else {
  analogWrite(motor11, vel);
  analogWrite(motor21, 0);
  analogWrite(motor12, 0);
  analogWrite(motor22, 0);
  delay(500);
}
```

Por último, el modo manual, activado cuando el Arduino recibe el carácter 'X', detiene los motores, y el usuario toma de nuevo el control del vehículo. Este modo puede activarse cuando el vehículo se encuentra desplazándose automáticamente.

```
if(estado=='X'){
     analogWrite(motor11, 0);
     analogWrite(motor21, 0);
     analogWrite(motor12, 0);
     analogWrite(motor22, 0);
```

5.2. Especificación del entorno de pruebas

El entorno de pruebas reúne todo lo necesario para realizar las pruebas del sistema, una vez finalizado el diseño y la implementación del mismo.

Para ello se utilizará un teléfono móvil y el prototipo del vehículo construido. Las características técnicas del dispositivo móvil utilizado son las siguientes:

- Memoria RAM de 2 GB
- Procesador HiSilicon Kirin 910T quad-core, 1.8 GHz, GPU Mali-450MP4
- Pantalla de 5 pulgadas, resolución 1080 x 1920 px
- Almacenamiento interno de 16 GB (extensible a 32GB)
- Cámara trasera de 13 megapíxeles. Cámara frontal de 8 megapíxeles.
- Conectividad LTE (4G)
- Sistema operativo Android 5.14

Ilustración 41: Huawei Ascend P7

5.3. Pruebas de funcionalidad

Identificador: PF_XX	Nombre de prueba funcional
Sistema	
Descripción	
Acciones	
Resultado	
Requisito de sistema	

Tabla 98: Formato tabla pruebas

Los campos que componen la tabla son:

- **Identificador**: identifica la prueba, y está formado por:
  - **PF**: abreviatura de Prueba Funcional, y común para todas las pruebas.
  - **XX**: número de la prueba. Será un número de dos cifras comprendido entre 01 y 99.
- **Nombre**: indica el título de la prueba funcional.
- **Sistema**: indica si la prueba es perteneciente al software o al hardware del sistema.
- **Descripción**: es la descripción con el objetivo de la prueba.
- **Acciones**: serie de pasos que deben seguirse, en el orden establecido, para llevar a cabo la prueba.
- **Resultado**: indica el resultado de la prueba. Puede ser:
  - **Válida**: la prueba ha sido superada con éxito.
  - **Inválida**: la prueba no ha sido superada, y es posible que se deban aplicar cambios al sistema.
- **Requisito de sistema**: indica el requisito de sistema al que hace referencia la prueba

PF_01	Activación del Bluetooth
Sistema	Software
Descripción	Verificación de que el usuario puede activar el Bluetooth desde la aplicación si lo tiene desactivado.
Acciones	1. Acceder la aplicación y pulsar en “Conectar”. 2. Pulsar “Aceptar” cuando aparezca el mensaje que ofrece al usuario activar el Bluetooth.
Resultado	Válido
Requisito de sistema	RS-F-01

Tabla 99: Prueba funcional PF_01

PF_02	Dispositivos enlazados
Sistema	Software
Descripción	Verificación de que el usuario no puede visualizar la lista de dispositivos enlazados con el teléfono si no tiene activado el Bluetooth.
Acciones	1. Acceder la aplicación y pulsar en “Conectar”. 2. Pulsar “Denegar” cuando aparezca al usuario activar el Bluetooth. 3. Verificar que no aparece ningún dispositivo enlazado en la lista. 4. Pulsar en “Ver dispositivos enlazados” y verificar que aparece un mensaje advirtiendo al usuario de que no existen dispositivos vinculados.
Resultado	Válido
Requisito de sistema (¿)	RS-F-02

Tabla 100: Prueba funcional PF_02

PF_03	Lista dispositivos enlazados
Sistema	Software
Descripción	Verificación de que el usuario puede visualizar en la lista de dispositivos enlazados el dispositivo Bluetooth del vehículo.
Acciones	1. Emparejar previamente el teléfono móvil con el dispositivo Bluetooth del vehículo (mediante contraseña). 2. Acceder a la aplicación y pulsar en “Conectar”. 3. Si no está activado el Bluetooth, activarlo pulsando en “Aceptar” en el mensaje ofrecido. Si está activado previamente, omitir este paso. 4. Pulsar en “Ver dispositivos enlazados”. 5. Verificar que el dispositivo del vehículo se encuentra en la lista.
Resultado	Válido
Requisito de sistema	RS-F-04

Tabla 101: Prueba funcional PF_03

PF_04	Conexión con dispositivo
Sistema	Software
Descripción	Verificación de que la conexión se ha establecido correctamente con el vehículo.
Acciones	1. Acceder la aplicación y pulsar en “Conectar”. 2. Pulsar en “Ver dispositivos enlazados”. 3. Verificar que el dispositivo del vehículo se encuentra en la lista y pulsar en él. 4. Redirección automática a la pantalla de control del vehículo.
Resultado	Válido
Requisito de sistema	RS-F-05

Tabla 102: Prueba funcional PF_04

PF_05	Reintentar conexión
Sistema	Software
Descripción	Verificación de que el usuario puede intentar conectarse de nuevo en caso de que el dispositivo no se encuentre disponible o la conexión falle.
Acciones	1. Acceder la aplicación y pulsar en “Conectar”. 2. Pulsar en “Ver dispositivos enlazados”. 3. Verificar que el dispositivo del vehículo se encuentra en la lista y pulsar en él. 4. Si la conexión no se produce, volver a iniciar la aplicación hasta que la conexión se realice.
Resultado	Válido
Requisito de sistema	RS-F-06

Tabla 103: Prueba funcional PF_05

PF_06	Manejo del vehículo manual
Sistema	Software
Descripción	Verificación de que el usuario puede manejar el vehículo mediante los botones correspondientes.
Acciones	1. Acceder la aplicación y pulsar en “Conectar”. 2. Pulsar en “Ver dispositivos enlazados”. 3. Verificar que el dispositivo del vehículo se encuentra en la lista y pulsar en él. 4. Redirección automática a la pantalla de control del vehículo. 5. Botones habilitados para el manejo manual.
Resultado	Válido
Requisito de sistema	RS-F-12

Tabla 104: Prueba funcional PF_06

PF_07	Versión de Android
Sistema	Software
Descripción	Verificación de que la aplicación funciona en la última versión de Android.
Acciones	1. Tener la versión 6.0 de Android instalada en el dispositivo. 2. Acceder a la aplicación. 3. Verificar que la aplicación se ejecuta con todas sus funcionalidades correctamente.
Resultado	Válido
Requisito de sistema	RS-NF-01

Tabla 105: Prueba funcional PF_07

PF_08	Visualización de la aplicación correcta
Sistema	Software
Descripción	Verificación de que la aplicación se visualiza correctamente en un Smartphone con un tamaño de 5 pulgadas.
Acciones	1. Disponer de un dispositivo inferior a 5.5 pulgadas. 2. Acceder a la aplicación. 3. Verificar que las interfaces de usuario se visualizan correctamente.
Resultado	Válido
Requisito de sistema	RS-NF-02

Tabla 106: Prueba funcional PF_08

PF_09	Idioma de la aplicación
Sistema	Software
Descripción	Verificación de que la aplicación se encuentra totalmente en castellano.
Acciones	1. Vincular los dispositivos. 2. Acceder a la aplicación. 3. Verificar que el lenguaje de la aplicación en su totalidad es el castellano.
Resultado	Válido
Requisito de sistema	RS-NF-03

Tabla 107: Prueba funcional PF_09

PF_10	Finalización de la aplicación
Sistema	Software
Descripción	Verificación de que el usuario puede finalizar la conexión Bluetooth con el vehículo y salir de la aplicación.
Acciones	1. Acceder la aplicación y pulsar en “Conectar”. 2. Pulsar en “Ver dispositivos enlazados”. 3. Verificar que el dispositivo del vehículo se encuentra en la lista y pulsar en él. 4. Redirección automática a la pantalla de control del vehículo. 5. Pulsar en “Desconectar” y verificar que la conexión finaliza y se sale de la aplicación.
Resultado	Válido
Requisito de sistema	RS-F-19

Tabla 108: Prueba funcional PF_10

PF_11	Teléfonos enlazados
Sistema	Software
Descripción	Verificación de que el dispositivo del vehículo permanece en la lista de dispositivos enlazados cuando se inicie de nuevo la aplicación.
Acciones	1. Cerrar la aplicación. 2. Acceder de nuevo la aplicación y pulsar en "Conectar". 3. Pulsar en "Ver dispositivos enlazados". 4. Verificar que el dispositivo del vehículo se encuentra en la lista.
Resultado	Válido
Requisito de sistema	RS-F-20

Tabla 109: Prueba funcional PF_11

PF_12	Vehículo enlazado con varios dispositivos
Sistema	Hardware
Descripción	Verificación de que el dispositivo del vehículo puede estar enlazado con uno o varios dispositivos.
Acciones	1. Tener disponible el módulo Bluetooth. 2. Emparejarlo con dos o más dispositivos móviles. 3. Verificar que aparece en las listas de dispositivos enlazados.
Resultado	Válido
Requisito de sistema	RS-F-24

Tabla 110: Prueba funcional PF_12

PF_13	Vehículo conectado con un dispositivo
Sistema	Hardware
Descripción	Verificación de que el dispositivo del vehículo puede estar conectado únicamente con un dispositivo.
Acciones	1. Tener el vehículo conectado a un dispositivo. 2. Acceder a la aplicación con otro dispositivo. 3. Buscar en la lista de dispositivos enlazado el del vehículo y pulsar en él. 4. Verificar que no se puede conectar ya que no se encuentra disponible.
Resultado	Válido
Requisito de sistema	RS-F-25

Tabla 111: Prueba funcional PF_13

PF_14	LED del módulo Bluetooth parpadea
Sistema	Hardware
Descripción	Verificación de que el LED del módulo Bluetooth parpadea si no está conectado a ningún dispositivo.
Acciones	1. Tener disponible el módulo Bluetooth (aplicar voltaje). 2. Verificar que el LED inicialmente parpadea.
Resultado	Válido
Requisito de sistema	RS-F-26

Tabla 112: Prueba funcional PF_14

PF_15	LED del módulo Bluetooth fijo
Sistema	Hardware
Descripción	Verificación de que el LED del módulo Bluetooth permanece fijo cuando se conecta a un dispositivo.
Acciones	1. Abrir la aplicación y conectarse a un dispositivo emparejado. 2. Esperar unos segundos. 3. Verificar que el LED ya no parpadea y permanece fijo.
Resultado	Válido
Requisito de sistema	RS-F-27

Tabla 113: Prueba funcional PF_15

PF_16	Sistema anti-colisiones
Sistema	Hardware
Descripción	Verificación de que el sistema anti-colisiones funciona correctamente en el modo automático.
Acciones	1. Acceder la aplicación y pulsar en “Conectar”. 2. Pulsar en “Ver dispositivos enlazados”. 3. Verificar que el dispositivo del vehículo se encuentra en la lista y pulsar en él. 4. Redirección automática a la pantalla de control del vehículo. 5. Pulsar en “Automático”. 6. Esperar a que el vehículo detecte un obstáculo. 7. Verificar que el vehículo no colisiona con el obstáculo y gira.
Resultado	Válido
Requisito de sistema	RS-F-30

Tabla 114: Prueba funcional PF_16

PF_17	Recepción correcta
Sistema	Hardware
Descripción	Verificación de que el Arduino recibe los datos del módulo Bluetooth correctamente
Acciones	1. Abrir la aplicación y conectarse a un dispositivo emparejado. 2. Manejar el vehículo y cambiar de modo. 3. Verificar que se realiza correctamente la operación
Resultado	Válido
Requisito de sistema	RS-F-36

Tabla 115: Prueba funcional PF_17

PF_18	Vehículo alimentado
Sistema	Hardware
Descripción	Verificación de que tanto el Arduino como los motores reciben alimentación externa de forma correcta.
Acciones	1. Verificar que el Arduino se enciende al conectarlo a la pila de 9V. 2. Verificar que los motores se mueven correctamente.
Resultado	Válido
Requisito de sistema	RS-F-36

Tabla 116: Prueba funcional PF_18

PF_19	Estructura vehículo estable
Sistema	Hardware
Descripción	Verificación de que la estructura del vehículo se encuentra estable y las ruedas lo desplazan correctamente.
Acciones	1. Mover el vehículo en modo manual o modo automático. 2. Verificar que la estructura se mantiene estable. 3. Verificar que las ruedas se mueven gracias a los motores y que la rueda loca puede moverse.
Resultado	Válido
Requisito de sistema	RS-F-36

Tabla 117: Prueba funcional PF_19

PF_20	Errores de la aplicación
Sistema	Software
Descripción	Todos los posibles errores existentes en la aplicación serán notificados al usuario.
Acciones	1. Probar la aplicación en un dispositivo que no soporte Bluetooth, por ejemplo, el emulador de Android Studio. 2. Verificar que se notifica el error correctamente.
Resultado	Válido
Requisito de sistema	RS-F-36

Tabla 118: Prueba funcional PF_20

5.4. Matriz de trazabilidad de pruebas

	PF01	PF02	PF03	PF04	PF05	PF06	PF07	PF08	PF09	PF10	PF11	PF12	PF13	PF14	PF15	PF16	PF17	PF18	PF19	PF20
RSF01	X																			
RSF02	X																			
RSF03		X																		
RSF04			X																	
RSF05				X																
RSF06				X																
RSF07					X															
RSF08					X															
RSF09						X														
RSF10						X														
RSF11						X														
RSF12						X														
RSF13						X														
RSF14						X														
RSF15						X														
RSF16						X														
RSF17						X														
RSF18						X														
RSF19						X														
RSF20											X									
RSF21																				x
RSF22										X										
RSF23																	X			
RSF24												X								
RSF25													X							
RSF26														X						
RSF27															X					
RSF28											X									
RSF29																	X			
RSF30																X				
RSF31																X				
RSF32																X				
RSF33																		X		
RSF34																			X	
RSF35																			X	
RSF36																	X			
RSF37																	X			
RSF38																	X			
RSF39																		X		
RSNF01							X													
RSNF02								X												
RSNF03									X											
RSNF04																	X			
RSNF05																	X			
RSNF06																	X			
RSNF07																	X			

Tabla 119: Matriz de trazabilidad de pruebas

Capítulo 6: Planificación del proyecto

En este capítulo se detalla la planificación llevada a cabo a lo largo de las distintas fases del proyecto, incluyendo las tareas con sus fechas estimadas de inicio y de fin, su duración total y un diagrama de Gantt con el desglose de las actividades en una línea de tiempo.

6.1. Planificación de tareas

La siguiente tabla muestra una planificación de las actividades llevadas a cabo en la realización del proyecto, con el tiempo asociado a cada una de ellas y la fecha de inicio y fin que tuvo lugar en cada actividad.

Actividad	Fecha inicio	Fecha fin	Duración (días)
Propuesta	07/03/2016	07/03/2016	1
Presupuesto y planificación	08/03/2016	15/03/2016	5
Viabilidad	16/03/2016	29/03/2016	11
Análisis	30/03/2016	19/04/2016	15
Diseño	20/04/2016	20/05/2016	26
Implementación	23/05/2016	15/08/2016	60
Pruebas	16/08/2016	23/08/2016	7
Documentación	24/08/2016	25/09/2016	30

Tabla 120: Planificación de tareas del proyecto

La fecha de inicio del proyecto es el día 7 de marzo de 2016, y la de finalización el 25 de septiembre de 2016. Aproximadamente el proyecto consta de aproximadamente 7 meses y medio de duración.

En el siguiente apartado se muestra la planificación de las actividades representada mediante un diagrama de Gantt.

6.2. Diagrama de Gantt

El siguiente diagrama de Gantt muestra el desglose de la planificación de las actividades a lo largo de la duración del proyecto.

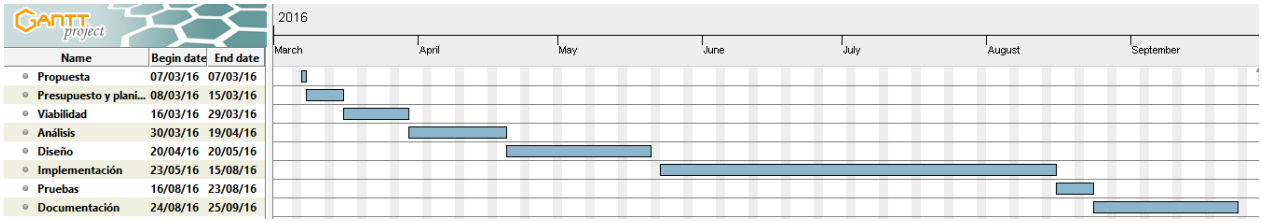


Ilustración 42: Diagrama de Gantt

Capítulo 7: Presupuesto del proyecto

El presupuesto del proyecto incluye un desglose de todos los costes que se deben tener en cuenta en la realización del proyecto, para poder determinar un coste total al cliente.

Teniendo en cuenta la planificación del proyecto descrita en el anterior apartado, se procede a establecer un presupuesto estimado para la realización del proyecto.

Todos los costes indicados en los siguientes apartados aparecen sin I.V.A. Para establecer la cantidad total, en el desglose de costes final, se incluirá el porcentaje correspondiente al I.V.A. de un 21%.

Para ello, se van a distinguir los diferentes costes en:

- **Costes de personal**: dependiendo del cargo desempeñado en el proyecto, cada persona tendrá asignado un número determinado de horas dedicadas al proyecto y un coste por hora distinto.

- **Costes de hardware**: en este proyecto la presencia del hardware es muy relevante, por lo que es necesario realizar un presupuesto de todos los componentes que se han utilizado, así como los equipos con los que se ha realizado el proyecto.

- **Costes de software**: para diseñar la parte de software, es posible que se haya necesitado disponer de licencias para determinados programas.

7.1. Resumen de horas dedicadas

En la siguiente tabla se muestra el desglose de las tareas realizadas a lo largo del proyecto y el total aproximado de horas dedicadas para cada una:

Actividad	Días	Horas/día	Total horas
Propuesta	1	6	6
Presupuesto y planificación	5	6	30
Viabilidad	11	8	88
Análisis	15	8	120
Diseño	26	8	208
Implementación	60	8	480
Pruebas	7	6	42
Documentación	30	6	180

Tabla 121: Resumen de horas dedicadas

Como se puede observar, el total de horas dedicadas al proyecto ha sido de 1154 horas aproximadamente.

7.2. Costes de personal

En este apartado se indica el desglose de costes de personal, diferenciando cada cargo realizado en el proyecto y el coste total.

Cargo	Nombre	Días	Horas / día	Total de horas	Salario / hora	Coste
Analista	Víctor Torrico López	14	8	200	18 €	3.600 €
Diseñador	Víctor Torrico López	30	8	240	20 €	4.800 €
Programador	Víctor Torrico López	75	6	600	15 €	9.000 €

Tabla 122: Costes de personal

El coste de personal asciende a un total de **17.400 euros** (sin I.V.A. incluido).

7.3. Costes de hardware

Para realizar el desglose de los costes de hardware, es necesario conocer:

- **Coste**: es el coste en euros, sin I.V.A., del componente hardware.
- **Porcentaje de uso en el proyecto**: indica el porcentaje en el que la componente ha sido usada en el proyecto.
- **Dedicación**: indica la cantidad de meses que es utilizada la componente hardware.
- **Deprecación**: indica la disminución periódica del valor de la componente. Para este proyecto se establece por defecto en 5 años, es decir, 60 meses.
- **Coste imputable**: es calculado como: (dedicación / deprecación) * coste equipo. Es el coste total en euros de la componente para el proyecto, también sin I.V.A.

La siguiente tabla resume los costes totales de las componentes:

Concepto	Coste	% Uso	Dedicación	Deprecación	Coste imputable
Ordenador portátil	950	100	7	60	110,83
Smartphone Android	250	100	7	60	29,16
Arduino UNO	25	100	7	-	25
Componentes Arduino	50	100	7	-	50
Estructura vehículo	25	100	7	-	25

Tabla 123: Costes de hardware

Tanto la placa Arduino UNO, como las componentes y el vehículo no se amortizan, dado que el uso es exclusivo para este proyecto. La placa Arduino UNO podría amortizarse a 5 años, como el portátil y el teléfono móvil, pero es la parte principal del vehículo, por lo que se ha decidido no amortizarlao.

El coste total del hardware para este proyecto es de **239,99 euros**.

7.4. Costes de software

En este apartado se detallan los costes en euros de las licencias de los programas necesarios para este proyecto.

Aquellos programas utilizados cuyo coste es gratuito no se incluyen.

Concepto	Coste
Microsoft Office 2016 "Home & Enterprise"	240
Windows 10	150

Tabla 124: Costes de software

El coste total del software para este proyecto asciende a un total de **390 euros**.

7.5. Resumen de costes

Concepto	Total (€)
Costes de personal	17.400,00
Costes de hardware	239,99
Costes de software	390,00
Costes indirectos (20%)	3.601,58
Coste total (sin I.V.A.)	21.631,57
TOTAL (I.V.A. 21% incl.)	**26.174,20**

Tabla 125: Resumen de costes

El coste total del proyecto sin I.V.A. asciende a la cantidad de 21.631,57 euros. Aplicando el porcentaje del 21% correspondiente al I.V.A., el precio final del proyecto es de **26.174,20 euros**.

Capítulo 8: Conclusiones y líneas futuras

En este capítulo se detallan las conclusiones tanto objetivas como subjetivas una vez finalizado el desarrollo del proyecto, y se incluyen una serie de líneas de trabajo que podrán ser llevadas a cabo en un futuro a partir de este proyecto.

8.1. Conclusiones del proyecto

El objetivo principal de este proyecto, que ha sido diseñar e implementar un prototipo de vehículo a escala para que pudiera ser controlado de forma remota a través de un dispositivo móvil, se puede afirmar que ha sido logrado de manera satisfactoria.

Para alcanzar este objetivo, se han tenido que cumplir previamente otros objetivos, definidos en el primer capítulo de este documento:

- Se ha conseguido diseñar y construir una estructura similar a la de un vehículo, en la que poder añadir toda la circuitería para que se mueva sin tener que tener una conexión a un ordenador o a la corriente eléctrica. La estructura no es un vehículo como tal, no contiene una carcasa que sea la de un vehículo radio control y es una estructura simple, pero el objetivo de este trabajo era crear un prototipo de vehículo, por lo que el aspecto externo resultante pasa a ser secundario.

- Se ha desarrollado una aplicación para Arduino en la que se tienen en cuenta tanto los posibles movimientos del vehículo como los dos modos de manejo. Se ha verificado que, con el envío de un carácter desde la aplicación del teléfono, mediante el botón correspondiente, el software cargado en la placa de Arduino recibe dicho carácter del módulo Bluetooth instalado en el vehículo y lo procesa, realizando en cada caso cierto movimiento de los motores o cambio de modo.

- Se ha desarrollado una aplicación el sistema operativo Android, cuya finalidad principal es manejar el vehículo. Gracias al diseño sencillo e intuitivo, uno de los principales objetivos del diseño de la aplicación, cualquier persona independientemente de la experiencia que tenga utilizando aplicaciones móviles, puede manejar el vehículo. Además, se adjunta un manual de usuario para facilitar el manejo de la aplicación al usuario.

El cumplimiento de dichos objetivos ha sido posible gracias al plan de análisis establecido, en el que se definieron todos los requisitos que se impusieron con el cliente. Los requisitos establecidos al principio han cumplido con las expectativas fundamentales del proyecto, y gracias a estos, han ido surgiendo otros que han ido añadiendo y modificando funcionalidades al sistema.

Una vez bien definidos y establecidos los requisitos del sistema, se comenzó a diseñar el sistema, tanto la parte de hardware como el software que contiene. Diseñar el hardware fue la tarea que llevó más tiempo, ya que son diversas las componentes que se han utilizado e integrarlas en la estructura del vehículo ha requerido dedicar más tiempo. Tanto el hardware como el software fue diseñado correctamente y posteriormente implementado por separado, para así poder disgregar las distintas tareas sin que las de un bloque interfiriesen en las de otro.

A lo largo de la realización del proyecto han surgido dificultades tanto en el software como en el hardware, que han podido modificar la planificación de las tareas establecida desde el principio; en el caso del software, los problemas habituales han sido errores en el código, tanto de Arduino como de la aplicación Android. Esto es algo muy habitual en la programación, por lo que, gracias a la documentación aportada por Internet y realizando numerosas pruebas, los errores fueron subsanados.

Las dificultades en el hardware han sido las más comunes en la implementación del sistema, como fallos en alguna componente por una mala conexión o un voltaje erróneo, desgaste de las pilas al realizar numerosas pruebas, piezas sueltas en la estructura del vehículo o la sustitución que se realizó de dos componentes al estropearse.

Los problemas mencionados, tanto en el software como en el hardware, originados durante la realización del proyecto, han podido ser solventados de manera eficiente y el plazo de entrega final del proyecto no se ha visto alterado, pese a sufrir alteraciones la planificación inicial de las tareas.

8.2. Conclusiones personales

La realización del proyecto, desde un primer momento, ha requerido un gran esfuerzo personal debido a la poca experiencia en el entorno del sistema.

Lo que ha resultado más sencillo, pero a la vez ha entrañado más problemas, ha sido la construcción del vehículo; todas las conexiones de las componentes hardware que constituyen la estructura pudieron realizarse correctamente gracias a la amplia documentación existente en Internet. Los problemas fueron sobre todo para acoplar la circuitería a la estructura del vehículo y la construcción de dicha estructura.

La parte más complicada, en la que se ha empleado mayor esfuerzo, ha sido en la programación del software, ya que contaba con poca experiencia programando aplicaciones Android y ninguna con Arduino.

Han sido numerosas las dificultades encontradas durante la realización del proyecto y han surgido contratiempos que han hecho que los plazos definidos para las tareas pudieran verse afectados.

8.3. Nuevas líneas de trabajo

En este apartado se exponen una serie de nuevas funcionalidades que pueden incorporarse al sistema teniendo en cuenta la funcionalidad actual del mismo.

- **Permitir al usuario modificar la velocidad a la que se desplaza el vehículo**

En la implementación actual del sistema, la velocidad es definida en el software de Arduino como una variable constante desde el momento en que se inicia la aplicación.

La idea consiste en añadir a la aplicación una opción para que el usuario pueda decidir la velocidad a la que se desplaza el vehículo en todo momento.

Esto podría implementarse bien mediante botones que incrementen o decrementen la velocidad, mediante una barra de desplazamiento o mediante un joystick. En cada caso se enviaría por el puerto serial el valor de la velocidad, y se asignaría a la variable que controla la velocidad de los motores en el programa de Arduino.

- **Permitir que el vehículo gire con distintos ángulos**

El sistema de giro implementado consiste en detener un motor durante un breve instante de tiempo mientras el otro se desplaza hacia adelante, para que se produzca un leve giro hacia el lado que el usuario desee.

De esta manera se está definiendo un ángulo de giro por defecto, de tal forma que si el usuario desea un giro más pronunciado, debe pulsar varias veces en el botón de giro.

La idea es similar a la descrita anteriormente relativa a la velocidad del vehículo, crear una barra de desplazamiento o un joystick de tal forma que el usuario realice el giro de manera más intuitiva. Para este caso, la opción de crear un joystick sería la más acertada.

- **Control del vehículo por Wi-Fi**

La principal limitación de usar la conectividad Bluetooth es el bajo rango de distancia que ofrece. A medida que el vehículo se aleja del teléfono móvil la conexión puede generar errores e incluso ser interrumpida.

Existen módulos que ofrecen conectividad Wi-Fi para conexiones de un dispositivo que disponga de Wi-Fi, como un ordenador, un Smartphone o una Tablet. Incorporando uno de estos módulos, como el módulo Wi-Fi ESP8266, se podría realizar el manejo del vehículo via Wi-Fi.

- **Implementar la aplicación para el control del vehículo compatible para otros sistemas operativos y dispositivos**

Al igual que la aplicación se ha programado para dispositivos Android, se podría implementar en entornos iOS o Windows Phone, lo que supone utilizar otro framework y otro lenguaje de programación.

También se podría adaptar la aplicación a otros dispositivos, como tablets, en el caso de Android, o iPads en el caso de iOS. La aplicación actual está diseñada para teléfonos móviles de un tamaño entre 3.5 y 5.5 pulgadas, y a pesar de poder ser instalada en tablets Android, no se visualizaría correctamente.

- **Dotar al vehículo de posicionamiento GPS**

Esto puede ser interesante por muchos motivos, por ejemplo, dotar al modo automático de un sistema de inteligencia artificial para detectar posiciones por las que ya ha pasado, y así evitar que se mueva más de una vez por la misma posición. Esto puede ser útil, por ejemplo, en el diseño de robots destinados al rastreo de una

posición en la búsqueda de algo concreto o en robots que limpian automáticamente una casa sin pasar dos veces por el mismo sitio.

Otro motivo por el que puede resultar interesante el posicionamiento GPS en el vehículo es para que se desplace automáticamente de un punto a otro, marcado por unas coordenadas GPS. Esto podría ser añadido y mejoraría el modo automático implementado en este proyecto, ya que en el estado actual se mueve automáticamente evitando obstáculos, pero sin un destino fijo.

- **Vehículo seguidor de línea**

Incorporando un sensor óptico compatible para Arduino e implementando la correspondiente funcionalidad, el vehículo podría seguir la trayectoria marcada por una línea situada en el suelo.

Anexo I: Glosario de términos

Definiciones

Bluetooth: conectividad que permite la transmisión de voz y datos entre diferentes dispositivos mediante un enlace por radiofrecuencia.

Chip: también denominado circuito integrado, es una estructura de pequeñas dimensiones sobre la que se fabrican circuitos electrónicos.

Dropbox: servicio de alojamiento de archivos multimedia en la nube, que permite sincronizar y almacenar archivos tanto privados como compartidos con otros usuarios.

Emulador: software que permite ejecutar un programa en una plataforma diferente a la original.

Hardware: conjunto de elementos físicos y materiales que componen un ordenador o un sistema informático.

Interfaz de usuario: medio con el que el usuario puede comunicarse e interactuar con una máquina, equipo o computadora, y comprende todos los puntos de contacto entre el usuario y el equipo.

Microcontrolador: circuito integrado programable, capaz de ejecutar órdenes contenidas en su memoria.

Microprocesador: procesador de dimensiones más pequeñas donde se realizan procesos lógicos.

One Drive: servicio de alojamiento de archivos creado por Microsoft con mejores prestaciones que Dropbox.

Sensor: objeto capaz de detectar magnitudes físicas o químicas para poder transformarlas en variables eléctricas.

Sistema operativo: conjunto de órdenes y programas que controlan los procesos básicos de una computadora, y permiten el funcionamiento de otros programas.

Skype: software que permite comunicaciones de voz, texto y vídeo por Internet.

Smartphone: teléfono móvil inteligente con prestaciones similares a las de un computador pequeño, con mayor conectividad que un teléfono móvil convencional y generalmente con una pantalla táctil.

Socket: concepto abstracto por el cual dos programas en computadoras distintas pueden intercambiar un flujo de datos, de manera fiable y ordenada.

Software: conjunto de programas que permiten a un ordenador o a un sistema realizar determinadas tareas.

Tablet: computadora portátil de mayor tamaño que un teléfono inteligente.

Ultrasonido: onda cuya frecuencia está por encima de la capacidad de audición del oído humano.

Acrónimos

DC: Corriente continua.

E/S: Entrada/Salida.

GB: Gigabyte (1024 Megabytes).

GHz: Gigahercio.

GPS: Global Positioning System (Sistema de posicionamiento global).

IVA: Impuesto sobre el Valor Añadido.

KB: Kilobyte (1024 bytes).

mA: Miliamperios.

MB: Megabyte (1024 Kilobytes).

MHz: Megahercio.

RAM: Random Access Memory (Memoria de acceso aleatorio).

SDK: Software Development Kit (Herramientas de desarrollo de software).

Wi-Fi: Wireless fidelity (Fidelidad inalámbrica).

Anexo II: Manual de usuario de la aplicación

Paso 1: Descarga e instalación de la aplicación

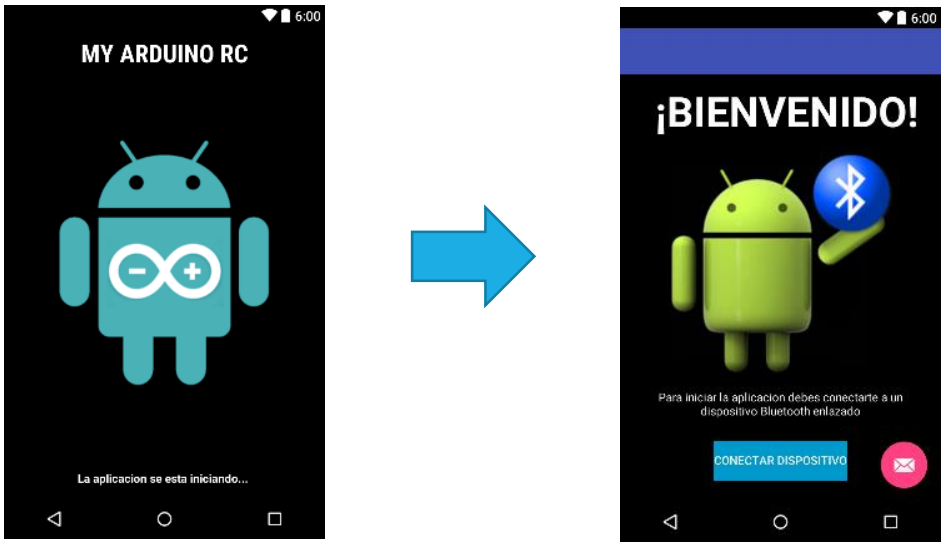
La aplicación se encuentra disponible en Google Play, la tienda oficial para descarga de aplicaciones Android. Por defecto, se encuentra instalada en cualquier dispositivo Android.

En la barra de búsqueda introducimos el nombre de la aplicación, My Arduino RC

Una vez encontrada, pulsamos en el botón Instalar, y se instalará automáticamente. La aplicación ocupa 2MB, por lo que no habrá problema en la instalación.

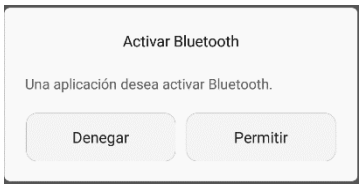
Paso 2: Inicio de la aplicación y conexión de dispositivo

Al iniciar la aplicación aparecerá la primera pantalla, que trascurrido un tiempo dará paso a la siguiente, donde aparece un texto de bienvenida y pide conectar un dispositivo.

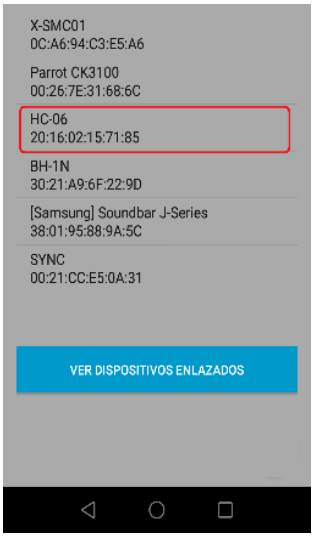


Al pinchar sobre el botón “Conectar Dispositivo”, nos conectaremos a un dispositivo enlazado con el teléfono. Si no se encuentra enlazado previamente, no aparecerá en la lista. Al final del manual se explica cómo enlazar dos dispositivos mediante Bluetooth.

Si no tenemos el Bluetooth activado, aparecerá el siguiente mensaje:



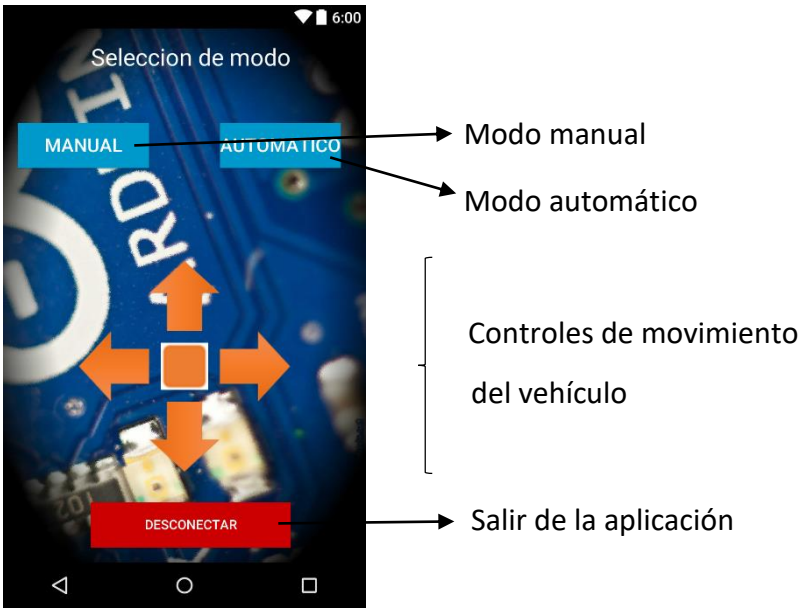
Pulsando en “Permitir”, se activará el Bluetooth y podrá visualizarse la lista de dispositivos vinculados pulsando sobre el botón “Ver dispositivos enlazados”.



En este caso, el dispositivo del vehículo vinculado al teléfono es el nombrado como HC-06, pero hay que seleccionar aquel que se haya enlazado con el teléfono previamente, y automáticamente se conectará con el vehículo.

Paso 3: Control del vehículo

Una vez realizada la conexión con éxito, ya se puede manejar el vehículo. La pantalla principal es la siguiente:



Para avanzar o retroceder el vehículo y detenerlo, es suficiente con pulsar una vez el botón.

Para realizar los giros se podrá pulsar el botón correspondiente las veces que se desee; el vehículo realizará pequeños giros en la dirección deseada y retomará la marcha hacia adelante.

Para cambiar el modo, se pulsará el botón de modo deseado.

Para salir de la aplicación, se deberá pulsar el botón “Desconectar”.

Enlazar dispositivos Bluetooth

Para conectarse a un dispositivo Bluetooth, se debe activar la conectividad Bluetooth del teléfono, y el dispositivo con el que enlazarse debe encontrarse visible.

En el caso del vehículo, el módulo del Bluetooth (HC-06) debe encontrarse con la luz verde parpadeando; esto significa que está visible y disponible para el enlace.

Los pasos para activar el Bluetooth son los siguientes:

1. Abrir el menú de Ajustes del teléfono.
2. Elegir “Bluetooth” en el menú de “Conexiones inalámbricas” o “Redes inalámbricas”.
3. Tocar la opción de Activar.
4. Una vez activado, en la barra de notificaciones (parte superior de la pantalla) aparecerá el siguiente icono .

El siguiente paso es conectarse a un dispositivo. Para ello:

1. El Bluetooth debe estar activado. Si no lo está, ver pasos anteriores.
2. En “Dispositivos disponibles” se encuentran todos aquellos dispositivos con los que poder establecer una conexión.
3. Pulsar sobre el dispositivo deseado (en el caso del vehículo, es necesario consultar previamente el nombre del módulo Bluetooth; el usado en el proyecto es HC-06).
4. Si requiere contraseña, probar a introducir 0000 o 1234, las más habituales. En caso de que no funcione con ninguna de ellas, consultar la documentación del dispositivo.

5. Si la contraseña es correcta, los dispositivos permanecerán enlazados correctamente.

Material bibliográfico

[1] Raspberry Pi Project.

https://www.raspberrypi.org/

[2] Características técnicas Raspberry Pi.

https://es.wikipedia.org/wiki/Raspberry_Pi

[3] Arduino.

https://es.wikipedia.org/wiki/Arduino

[4] Arduino UNO.

https://www.arduino.cc/en/Main/ArduinoBoardUno

[5] Arduino Mega.

https://www.arduino.cc/en/Main/arduinoBoardMega

[6] Uzebox Project.

http://obsoletos.org/2009/04/uzebox-la-consola-que-te-haces-tu-mismo/

[7] Comparativa entre Raspberry Pi y Cubieboard.

http://www.xatakahome.com/trucos-y-bricolaje-smart/cubieboard-primeras-impresiones-del-rival-de-la-raspberry-pi

[8] RepRap Project.

https://es.wikipedia.org/wiki/Proyecto_RepRap

[9] Giro de un motor de corriente continua.

http://diymakers.es/control-velocidad-y-sentido-de-motor-dc/

[10] Puente H Arduino.

http://panamahitek.com/el-puente-h-invirtiendo-el-sentido-de-giro-de-un-motor-con-arduino/

[11] Configuración chip L293D.

http://www.instructables.com/id/Control-your-motors-with-L293D-and-Arduino/

[12] Configuración Módulo Bluetooth HC-06.

http://www.prometec.net/bt-hc06/

[13] Tutorial configuración HC-SR04.

http://www.geekfactory.mx/tutoriales/tutoriales-arduino/sensor-ultrasonico-hc-sr04-y-arduino/

[14] Tutorial uso MIT App Inventor 2.

http://appinventor.mit.edu/explore/tutorial-version/app-inventor-2.html

[15] Herramienta Minibloq.

http://blog.minibloq.org/2012/01/nuevo-tutorial-con-ejemplos-en-espanol.html

[16] Entornos gráficos para desarrollo de Arduino.

http://robologs.net/2013/11/05/5-entornos-graficos-para-arduino/

[17] Entornos gráficos para desarrollo de aplicaciones Android.
http://www.pcactual.com/noticias/trucos/atrevete-crear-propias-aplicaciones-para-android-2_11544

[18] Herramienta Gantt Project.

http://www.ganttproject.biz/

[19] Eclipse (Software).

https://es.wikipedia.org/wiki/Eclipse_(software)

[20] Programación de sockets Bluetooth en Android Studio.

https://developer.android.com/reference/android/bluetooth/BluetoothSocket.html

[21] Doble puente H en Arduino.

https://prezi.com/8aznbsvshbv3/circuito-integrado-l293b-doble-puente-h/

[22] Configuración módulos Bluetooth HC-05 y HC-06

http://www.geekfactory.mx/tutoriales/bluetooth-hc-05-y-hc-06-tutorial-de-configuracion/

[23] Hardware libre (open hardware).

https://es.wikipedia.org/wiki/Hardware_libre

[24] Funcionamiento buffer triestado.

https://es.wikipedia.org/wiki/Buffer_triestado

[25] Motores con L293D

http://robots-argentina.com.ar/MotorCC_L293D.htm

[26] Productos oficiales Arduino.

https://www.arduino.cc/en/Main/Products

[27] Windows Phone.

https://es.wikipedia.org/wiki/Windows_Phone

[28] iOS.

https://es.wikipedia.org/wiki/IOS

[29] Android.

https://es.wikipedia.org/wiki/Android

[30] Cuota de mercado de telefonía móvil en España.

http://cincodias.com/cincodias/2016/04/06/smartphones/1459957587_678421.html

[31] Tienda online Aliexpress.

https://es.aliexpress.com/es_home.htm

[32] Enlazar dispositivos Bluetooth.

https://support.google.com/nexus/answer/2819579?hl=es

[33] Arduino IDE.

http://playground.arduino.cc/Es/OSW06